Un Framework Basé Bigraphes pour la Conception et l'Analyse des Systèmes Sensibles au Contexte

Taha Abdelmoutaleb Cherfia

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Constantine 2 - Abdelhamid Mehri
Faculté des Nouvelles Technologies de l'Information et de la Communication
Département de Technologies des Logiciels et des Systèmes d'Information

Un Framework Basé Bigraphes pour la Conception et l'Analyse des Systèmes Sensibles au Contexte

Année : 2016
N° d'ordre :
Série :
Pour l'obtention du grade de Docteur en 3ème cycle LMD
Spécialité : Informatique
Option : Génie Logiciel
Par

Taha Abdelmoutaleb CHERFIA

Soutenance prévue le 06/01/2016 devant la commission d'examen

Prof. Zizette BOUFAIDA	Université Constantine 2	Présidente du jury
Prof. Faïza BELALA	Université Constantine 2	Directrice de thèse
Prof. Thouraya BOUABANA-TEBIBEL	École Nationale Supérieure d'Informatique	Examinatrice
Prof. Djamel-Eddine SAÏDOUNI	Université Constantine 2	Examinateur
Dr. Saber BENHARZALLAH	Université Mohamed Khider de Biskra	Examinateur
Dr. Nabil HAMEURLAIN	Université de Pau et des Pays de l'Adour	Invité

Laboratoire d'Informatique Répartie (LIRE)
Constantine, Algérie

Dédicace

Á mes chers parents

Remerciements

Je tiens, tout d'abord, à exprimer mes plus vifs remerciements à ma directrice de thèse, Madame Faïza BELALA, professeur au département des Technologies des Logiciels et des Systèmes d'Information (TLSI) de l'Université Constantine 2 (Abdelhamid Mehri), pour avoir accepté de diriger mes travaux de recherche. Je la remercie pour ses précieux conseils, sa disponibilité et sa patience depuis mes premiers pas jusqu'à l'intégration totale dans la recherche. Son expérience et sa motivation ont rendu possible les contributions scientifiques qui constituent l'ossature de cette thèse.

Un immense merci à Monsieur Nabil HAMEURLAIN, maître de conférences habilité à diriger des recherches à l'Université de Pau et des Pays de l'Adour (UPPA) de m'avoir accueilli dans son équipe MOVIES (Modélisation, Visualisation, Exécution et Simulation), ainsi que pour son suivi pendant toute la durée de mon stage.

Je tiens à remercier Madame Zizette BOUFAIDA, professeur au département des Technologies des Logiciels et des Systèmes d'Information (TLSI) de l'Université Constantine 2 (Abdelhamid Mehri), pour m'avoir fait l'honneur d'accepter de présider le jury de ma thèse.

Je suis particulièrement sensible, par ailleurs, au grand honneur dont m'ont gratifié Professeur Thouraya BOUABANA-TEBIBEL de l'Ecole Nationale Supérieure d'Informatique, Professeur Djamel-Eddine SAÏDOUNI de l'Université Constantine 2 (Abdelhamid Mehri), et Monsieur Saber BENHARZALLAH, maître de conférences à l'Université Mohamed Khider de Biskra, pour l'intérêt qu'ils ont bien voulu porter à mes travaux en acceptant d'en être les rapporteurs, qu'ils trouvent ici l'expression de ma profonde reconnaissance.

Je tiens également à remercier tous mes amis et les membres du laboratoire LIRE (Laboratoire d'informatique répartie) de l'Université Constantine 2 (Abdelhamid Mehri), pour leurs conseils et leur soutien, qui a fait de cette maîtrise une expérience enrichissante et agréable.

En particulier, mes remerciement vont également à l'équipe GLSD (Génie Logiciel et Systèmes Distribués), animée par des doctorants agréables à côtoyer et passionnés par la recherche. Je retiendrai la bonne humeur des réunions d'équipe ou des présentation des exposés.

Finalement, j'adresse mes remerciements infinis à ma famille et surtout mes parents qui mon soutenu tout au long de mes études. Je les remercie pour leurs encouragements, leur dévouement et leur soutien inconditionnel durant toutes ces années.

Résumé

Aujourd'hui, les nouvelles technologies font partie de notre vie quotidienne. Qu'il s'agisse de s'informer, de se divertir, ou même de communiquer avec ses amis, les possibilités qu'offre l'informatique ubiquitaire sont innombrables. Pour que ces possibilités puissent devenir une réalité, les systèmes informatiques doivent alors se doter d'une capacité d'observation de leur environnement et de s'adapter en fonction des attentes et des besoins des utilisateurs. C'est ce qu'on appelle la sensibilité au contexte. En effet, la littérature montre que la sensibilité au contexte est le point central de l'informatique ubiquitaire. Cependant, face à l'hétérogénéité et la dynamicité des informations de contexte, sa prise en compte nécessite la mise en place d'un modèle pour décrire ces informations à un haut niveau d'abstraction.

Dans ce travail de thèse nous proposons, dans un premier temps, un modèle appelé BigCAS (Bigraphical Contexte-Aware System) qui permet la conception formelle des systèmes sensibles au contexte. Pour accomplir cet objectif, BigCAS repose sur des modèles formelles à base des systèmes réactifs bigraphiques permettant la modélisation des aspects structurels et comportementaux des systèmes sensibles au contexte. Il offre une séparation claire entre la partie sensible au contexte et la partie non-sensible au contexte de ces systèmes. Chacune de ces parties est modélisée séparément par un bigraphe distinct, où la composition de ceux-ci modélise la structure générale du système ainsi que les interactions et les effets de bord entre ses différents composants. Par ailleurs, BigCAS tient compte non seulement des aspects structurels, mais aussi des différentes reconfigurations intervenant dans le comportement des systèmes sensibles au contexte.

Nous proposons également une extension du modèle BigCAS, appelée BigCAS-FA (Bigraphical Context-Aware System - Formal Analysis), qui permet la vérification formelle de propriétés de sûreté et de vivacité des systèmes sensibles au contexte. En outre, BigCAS-FA possède des stratégies à base de contrats qui consistent à guider la reconfiguration dynamique en fonction du contexte.

Afin de valider nos propositions, nous développons le prototype BigCAS-Tool (Bigraphical Context Aware System - Tool) dédié à la spécification et la vérification des systèmes sensibles au contexte, et nous l'illustrons à travers une étude de cas d'un système d'éclairage intelligent.

Mots-clés : Informatique Ubiquitaire, Systèmes Sensibles au Contexte, Systèmes Réactifs Bigraphiques, BigCAS, BigCAS-FA, BigCAS-Tool

Abstract

Today, modern technologies have become a part of our daily life. Whether to be informed, entertained, or even to communicate with friends, ubiquitous computing offers numerous opportunities. For this to become reality, computer systems must be able to observe the environment and to adapt their behaviour according to the users expectations and needs. This is called context-awareness. Indeed, the literature shows that context-awareness is the focal point of ubiquitous computing. However, due to heterogeneity and dynamicity of context information, taking it into account requires establishing a model to represent these information at a high-level of abstraction.

In this thesis, we propose a model called BigCAS (Bigraphical Context-Aware System) that supports the design of context-aware systems. To achieve this goal, BigCAS is based on formal specifications, derived from bigraphical reactive systems, for modelling structural and behavioural aspects of context aware systems. It provides a clear separation between the context-aware part and the context-unaware part of these systems. Each part is modelled separately as a distinct bigraph, where the composition of these bigraphs models the general structure of the system, particularly, its components interactions and their side effects. Moreover, BigCAS considers not only structural aspects, but also the different reconfigurations involved in the behaviour of context aware systems.

We also propose an extension to BigCAS, named BigCAS-FA (Bigraphical Context-Aware System - Formal Analysis), that provides formal verification of safety and liveness properties of context aware systems. Furthermore, BigCAS-FA provides contract-based strategies to guide the dynamic reconfiguration according to the context.

To validate our proposals, we develop a prototype, BigCAS-Tool (Bigraphical Context Aware System - Tool), devoted to the specification and verification of context-aware systems. The proposed prototype is illustrated with a case study of a smart lighting system.

Keywords : Ubiquitous Computing, Context-Aware Systems, Bigraphical Reactive Systems, BigCAS, BigCAS-FA, BigCAS-Tool

ملخص

اليوم، أصبحت التقنيات الحديثة جزءا من حياتنا اليومية. سواء في الإعلام ، الترفيه، أو حتى التواصل مع الأصدقاء، الحوسبة في كل مكان تتيح فرصا كثيرة. لكي تصبح هذه الفرص حقيقة واقعة، يجب أن تكون أنظمة الكمبيوتر قادرة على مراقبة البيئة المحيطة بها وأن تكييف وفقا لتوقعات المستخدمين واحتياجاتهم. وهذا ما يسمى بسياق الوعي.

في الواقع، يظهر الأدب أن سياق الوعي هو النقطة المحورية في الحوسبة في كل مكان. ولكن نظرا لعدم التجانس وديناميكية المعلومات، يتطلب إدراج سياق الوعي إنشاء نموذج لتمثيل هذه المعلومات على مستوى عال من التجريد.

في هذه الأطروحة ، نقترح كخطوة أولى نموذج يسمى BigCAS (Bigraphical Context-Aware System) والذي يدعم تصميم أنظمة سياق الوعي. لتحقيق هذا الهدف، يستند BigCAS على مواصفات رسمية مستمدة من bigraphical reactive systems، لنمذجة الجوانب الهيكلية والسلوكية لهذه الأنظمة. النموذج المقترح يوفر فصلا واضحا بين الجزء الواعي بالسياق و الجزء الغير الواعي من هذه النظم. كل جزء يتم تصميمه على حدا على شكل bigraph، بحيث يكون ذمج هذه الأشكال نموذجا للهيكل العام للنظام. علاوة على ذلك، لا تتوقف مهمة النوذج المقترح على تصميم الجوانب الهيكلية فحسب، ولكن يقوم أيضا بتصميم مختلف سلوكيات هذه الأنظمة.

كما نقترح أيضا ملحقا للنموذج BigCAS يدعى BigCAS-FA(Bigraphical Context-Aware System – Fomral Analysis) والذي يسمح بالتحقق الرسمي من خصائص السلامة و الحيوية لأنظمة سياق الوعي. بالإضافة إلى ذلك، فإن BigCAS-FA يتبنى استراتيجيات قائمة على عقود لتوجيه إعادة حيوية هذه النظم وفقا للسياق.

للتحقق من صحة مقترحاتنا، نقوم بتطويرنموذج أولي يسمى BigCAS-Tool(Bigraphical Context-Aware System – Tool) والمخصص لتصميم والتحقق من أنظمة سياق الوعي، ثم نقوم بإيضاحه عن طريق دراسة حالة لنظام الإضاءة الذكية.

الكلمات المفتاحية : الحوسبة في كل مكان، أنظمة وعي السياق، Bigraphical Reactive Systems، BigCAS، BigCAS-FA، BigCAS-Tool

Publications

Revues internationales

— Taha Abdelmoutaleb Cherfia and Faïza Belala. Bigraphical reactive systems based approaches for modeling context-aware systems. *International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS)*, 5(4):1–19, 2014

Conférences internationales

— Taha Abdelmoutaleb Cherfia, Kamel Barkaoui, and Faïza Belala. A brs-based modeling approach for context-aware systems: A case study of smart car system. In *Embedded and Ubiquitous Computing (EUC), 2014 12th IEEE International Conference on*, pages 310–314. IEEE, 2014

— Taha Abdelmoutaleb Cherfia and Faïza Belala. Towards a bigraph-based model for context-aware adaptive systems. In *Software Architecture*, pages 340–343. Springer, 2013

Workshops internationaux

— Taha Abdelmoutaleb Cherfia, Faiza Belala, and Kamel Barkaoui. Towards formal modeling and verification of context-aware systems. In *Proceedings of the 8th International Workshop on Verification and Evaluation of Computer and Communication Systems, VECoS 2014, Bejaïa, Algeria, September 29-30, 2014.*, pages 18–24, 2014

— Taha A Cherfia, Belala Faïza, and Nadira Benlahrache. Modeling of architectural reconfiguration case study: Automated teller machine. In *Advanced Information Systems for Enterprises (IWAISE), 2012 Second International Workshop on*, pages 40–46. IEEE, 2012

Abréviations

ACP	Algebra of Communicating Processes
API	Application Program Interface
ARAN	Authenticated Routing for Ad hoc Networks
BDI	Belief Desire Intention
BiAgent	Bigraphical Agent
BigCAS	Bigraphical Context-Aware Systems
BigCAS-FA	Bigraphical Context-Aware Systems Formal Analysis
BigCAS-Tool	Bigraphical Context-Aware Systems Tool
BigMC	Bigraphical Model Checker
BigraphER	Bigraph Evaluator and Rewriting
Big Red	Bigraph Editor
BPL	Bigraphical Programming Language Project
BPL-Tool	Bigraphical Programming Language Project Tool
BRS	Bigraphical Reactive Systems
CCS	Calculus of Communicating Systems
CHAM	Chemical Abstract Machine
CML	Context Modeling Language
COBRA-ONT	Context Broker Architecture ontology
CONON	Context Ontology
COOL	Context Ontology Language
ContextUML	Context Unified Modeling Language
CSP	Communicating Sequential Processes
CTMC	Continuous-Time Markov Chain
DbC	Design by Contract
DSL	Domain-Specific Language
EMF	Eclipse Modeling Framework
EXSD	Extension XML Schema Definition
F-Logic	Frame Logic
GEF	Graphical Editing Framework
GMF	Graphical Modeling Framework

Graphviz	Graph Visualization Software
GPS	Global Positioning System
GSM	Global System for Mobile Communications
IDE	Integrated Development Environment
IDL	Interface Description Language
JML	Java Modeling Language
JVM	Java Virtual Machine
LaCoMoCo	Laboratory for Context-dependentMobile Communication
MAC	Medium Access Control
MiniSAT	minimalistic Satisfability
MLContext	Modeling Language for Context-Aware Systems
MOF	Meta Object Facility
OCaml	Objective Categorical Abstract Machine Language
OCL	Object Constraint Language
OMG	Object Management Group
ORM	Object-Role Modeling
OSGI	Open Services Gateway initiative
OWL	Web Ontology Language
OWL-DL	Web Ontology Language Description Logic
PDA	Personal Digital Assitant
QoS	Quality of Service
RDF	Resource Description Framework
SAT	Satisfability
SML	Standard Meta Language
SWT	Standard Widget Toolkit
UML	Unified Modeling Language
Wi-Fi	Wireless Fidelity
XML	Extensible Markup Language

Table des matières

Table des figures

Liste des tableaux

Liste des listings

Introduction

De nos jours, l'informatique ne cesse d'évoluer et la technologie devient de plus en plus accessible à tout un chacun. Nous sommes passés du stade de l'ordinateur par foyer à celui de plusieurs ordinateurs pour une seule personne. Les ordinateurs sont devenus plus petits, plus puissants et incontournables de la vie quotidienne. Grâce à la prolifération des équipements informatiques et aux technologies sans fils, les ordinateurs sont intégrés partout dans l'environnement, échangeant des informations entre eux afin de rendre des services de manière transparente pour l'utilisateur. Nous parlons dans ce cas d'une informatique invisible et présente en tout lieu et à tout instant. C'est en partant de cette idée que l'informatique ubiquitaire est née.

Contexte de la thèse

À l'origine, l'informatique ubiquitaire a été introduite en 1991 par Mark Weiser dans le projet Xerox PARC [Wei91] pour décrire sa vision futuriste de l'ordinateur du 21e siècle. Il a déclaré que « les technologies les plus profondes sont celles qui disparaissent : elles se mêlent à l'élaboration de notre vie quotidienne jusqu'à en devenir indiscernables » [1]. L'idée novatrice de cette vision réside dans le fait de rendre l'information accessible à l'utilisateur n'importe où, n'importe quand et à partir de n'importe quel dispositif. Contrairement à l'informatique traditionnelle où un ordinateur central est associé à plusieurs individus, dans l'informatique ubiquitaire, la technologie se fond dans l'environnement afin de faire communiquer plusieurs ordinateurs pour fournir des services personnalisés à une seule personne.

Aujourd'hui, la vision de Mark Weiser est concrétisée grâce à la miniaturisation des composants électroniques (ordinateurs portables, smartphones tablettes, PDAs, etc), l'omniprésence des technologies réseaux sans fil (Wi-Fi, bluetooth, 802.15.4, GSM, 3G/4G, etc) et la démocratisation de la technologie et son intégration dans la vie quotidienne. Comme le souligne Gérard Berry dans son livre « Pourquoi et comment le monde devient numérique » [Ber08],

1. The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.

« les téléphones, appareils photos et caméras vidéo, les lecteurs et instruments de musique, les contrôleurs enfouis dans les avions, les voitures ou encore l'électroménager deviennent des ordinateurs habillés autrement ».

Dans la littérature, les termes : informatique ubiquitaire, informatique pervasive et intelligence ambiante sont souvent utilisés de façon indistincte. Le terme intelligence ambiante [ZEBD98] a été proposé lors d'un atelier organisé par Philips dont le but était le développement de scénarios pour l'utilisation des dispositifs ubiquitaires. Ensuite, Uwe Hansmann de la société IBM a proposé le terme informatique pervasive [Han03] pour décrire la technologie ubiquitaire. Cependant, malgré l'évolution du paradigme de l'informatique ubiquitaire, nous pouvons remarquer que la définition du contexte reste une préoccupation récurrente.

« Le contexte est toute information utilisée pour caractériser la situation d'une entité. Une entité peut être une personne, un lieu ou un objet considéré comme pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application même ». C'est avec ces mots que Dey [Dey01] a donné une définition au contexte, qui est devenue après la plus répandue et la plus adoptée dans les travaux portant sur la sensibilité au contexte, à ce jour.

La sensibilité au contexte est la propriété principale d'un système ubiquitaire. Elle caractérise la capacité d'un système à s'adapter aux changements du contexte. De ce fait, un système sensible au contexte est un système informatique capable de percevoir et d'utiliser les différentes informations relatives au contexte courant pour adapter dynamiquement sa fonctionnalité aux besoins de l'utilisateur.

Problématique

L'informatique sensible au contexte consiste à réduire la quantité d'informations explicitement fournies par l'utilisateur pour que le système accomplisse la tâche souhaitée. Aujourd'hui, les travaux de recherche accordent un grand intérêt à ce paradigme due au fait que la sensibilité au contexte semble être une solution répondant aux exigences issues de la mobilité dans un environnement en changement constant [SLP04]. Plusieurs domaines s'orientent de plus en plus vers l'informatique sensible au contexte pour proposer des services mobiles et intelligents qui nécessitent de moindres efforts de la part de leur utilisateur. Parmi ces domaines, nous citons par exemple la domotique [Ald03] où la maison est devenue un environnement ubiquitaire qui assiste les habitants dans leurs activités quotidiennes pour un meilleur confort.

Cependant, les informations de contexte sont généralement des informations brutes provenant de sources hétérogènes et distribuées. De ce fait, elles peuvent être sujets à des

ambiguïtés, des incohérences ou des incomplétudes. De plus, le contexte est un concept à caractère dynamique qui peut potentiellement varier d'une situation à une autre, ce qui apporte de nouvelles préoccupations lors de la conception et l'analyse des systèmes sensibles au contexte.

Afin de faciliter l'exploitation des informations de contexte, il est indispensable de fournir un niveau d'abstraction à ces informations. Cette abstraction peut être obtenue par la mise en place d'un modèle de description du contexte. Par conséquent, la modélisation est une tâche impérative dans la conception d'un système sensible au contexte qui nécessite la définition d'un modèle permettant de fournir une description abstraite des informations contextuelles. Ainsi le respect d'un modèle s'avère utile voire nécessaire.

Dans ce contexte, de nombreuses approches ont étés proposées dans la littérature afin de modéliser les informations de contexte. Ces approches sont basées sur différents modèles, à savoir les algèbres de processus, les paires clé-valeur, les langages de balisage, les modèles graphiques, les modèles orientés objet et les ontologies. Cependant, la majorité de ces approches focalisent leurs efforts soit à proposer des modèles pour décrire un contexte spécifique à une application ou à un domaine particulier, ou bien il leur manque le formalisme nécessaire pour la modélisation. De plus, l'hétérogénéité des informations de contexte nécessite l'usage de modèles génériques afin qu'ils soient adaptés à tout type de système.

Les systèmes réactifs bigraphiques semblent les mieux adaptés pour décrire des systèmes qui répondent aux besoins ubiquitaires qui sont la mobilité, la distribution et la dynamicité. En effet, l'un des principaux objectifs de la théorie des systèmes réactifs bigraphiques est d'intégrer dans un seul formalisme les aspects importants des systèmes ubiquitaires [Mil09]. Bien qu'il existe déjà quelques approches à base des systèmes réactifs bigraphiques visant à modéliser les systèmes sensibles au contexte, celles-ci ne prennent en compte que l'aspect statique lié à la structure alors que l'essence même de ces systèmes est dynamique. De plus, elles n'offrent aucun moyen de vérifier la validité de leurs modèles.

Objectifs et Contributions

Dans le cadre de cette thèse, nous nous intéressons à la modélisation et l'analyse des systèmes sensibles au contexte. Pour simplifier ces tâches, il est nécessaire d'utiliser des modèles formelles qui prennent en charge les aspects liés à la sensibilité au contexte. Par conséquent, notre proposition doit fournir un modèle générique qui permet, à la fois, la description des modèles représentatifs et vérifiables des systèmes sensibles au contexte.

Les contributions apportées par ce travail de recherche comportent trois volets :

Dans un premier temps, nous proposons un modèle formel à base de systèmes réactifs bigraphiques, nommé BigCAS (Bigraphical Context-Aware System), permettant la modélisation des aspects structurels et comportementaux des systèmes sensibles au contexte. Sur le plan structurel, ce modèle offre une séparation claire entre la partie du système dont la structure supporte certaine variabilité en fonction des changements éventuels du contexte, et la partie invariante vis-à-vis ces changements. Chacune de ces parties est modélisée séparément par un bigraphe distinct, un bigraphe sensible au contexte et un bigraphe non-sensible au contexte, respectivement. La composition de ces deux bigraphes permet de modéliser la structure générale du système. Par rapport aux approches orientées systèmes réactifs bigraphiques existantes, BigCAS permet non seulement la modélisation des aspects structurels, mais aussi la description des différentes reconfigurations internes et contextuelles intervenant dans le comportement des systèmes sensibles au contexte. Le modèle BigCAS sera illustré par une étude de cas portant sur un scénario de système d'éclairage dans une maison intelligente, où nous montrons son apport sur la modélisation des systèmes sensibles au contexte.

Pour l'analyse formelle des modèles de systèmes sensibles au contexte, nous proposons une extension de modèle BigCAS, appelée BigCAS-FA (Bigraphical Context-Aware System - Formal Analysis), qui permet la vérification des certaines propriétés relatives à la sûreté et à la vivacité de tels systèmes. Plus précisément, nous utilisons le model-checker BigMC (Bigraphical Model Checker) [PDH12]) pour vérifier des propriétés spécifiées sur un modèle de système sensible au contexte défini en BigCAS. En outre, BigCAS-FA intègre une stratégie de reconfiguration à base de contrats qui consiste à guider le processus d'adaptation au contexte.

Afin de valider nos différentes contributions, nous développons un prototype, nommé BigCAS-Tool (Bigraphical Context Aware System - Tool), permettant la modélisation et l'analyse des systèmes sensibles au contexte. BigCAS-Tool est une extension de l'outil Big Red [FPH13] qui consiste en un ensemble d'éditeurs graphiques mis à disposition pour la manipulation des modèles de bigraphes associés aux systèmes sensibles au contexte. De plus, il prend en charge les éléments essentiels de la modélisation et de la vérification, ce qui facilite ainsi à l'utilisateur la création des modèles des systèmes sensibles au contexte qui peuvent être vérifiés sans connaissance approfondie ou informations techniques. En effet, la tâche de l'utilisateur se résume uniquement à décrire le modèle associé à leurs systèmes sensibles au contexte en utilisant l'éditeur de bigraphes de BigCAS-Tool.

Organisation du manuscrit

Le manuscrit est structuré en six chapitres.

Le chapitre 1 est consacré à la présentation de l'informatique sensible au contexte qui

constitue le cadre général de notre recherche. Il présente une revue de littérature sur les différents concepts liés à ce thème et explore l'importance de la sensibilité au contexte pour les différents niveaux d'un système informatique : acquisition, représentation et adaptation au contexte.

Le chapitre 2 étudie les différentes approches que nous avons jugées intéressantes pour la modélisation des systèmes sensibles au contexte, à savoir les approches orientées algèbre de processus, les approches orientées modèle, les approches orientées ontologie, et les approches orientées systèmes réactifs bigraphiques. Il aborde les bases conceptuelles de ces approches, leur adéquation pour le support des caractéristiques des systèmes sensibles au contexte, leur pertinence et leurs limites. Puis, il présente une étude comparative entre ces différentes approches selon des critères bien précis.

Le chapitre 3 consiste en une étude bibliographique approfondie portant sur les systèmes réactifs bigraphiques. Il décrit l'anatomie des bigraphes, leurs caractéristiques, les principales opérations que nous pouvons effectuer sur eux. Puis, Il introduit la dynamique des systèmes réactifs bigraphiques ainsi que leur langage de termes algébriques. De plus, il présente les principaux outils de modélisation et simulation développés autour de ce formalisme.

Le chapitre 4 est consacré au premier objectif de cette thèse qui est la modélisation des systèmes sensibles au contexte. Il décrit notre vision d'un système sensible au contexte ainsi que notre approche orientée systèmes réactifs bigraphiques, appelée BigCAS (Bigraphical Context-Aware System), permettant la spécification des aspects structurels et comportementaux des systèmes sensibles au contexte. Puis, il présente des expérimentations de validation de l'approche proposée et illustre l'application de la démarche à travers un cas d'étude d'un système d'éclairage intelligent.

Le chapitre 5 est consacré au deuxième objectif de cette thèse qui est l'analyse des systèmes sensibles au contexte. Il présente une extension de notre modèle BigCAS, appelée BigCAS-FA (Bigraphical Context-Aware System - Formal Analysis), pour l'analyse formelle des propriétés intrinsèques des systèmes sensible au contexte.

Le chapitre 6 présente la mise en œuvre de l'outil BigCAS-Tool (Bigraphical Context-Aware System - Tool) implémentant notre modèle proposé. Il décrit l'architecture de l'outil ainsi que les différentes étapes de sa mise en œuvre. Puis, il décrit les résultats de l'expérimentation pour une validation applicative de notre modèle BigCAS.

Enfin, le manuscrit se termine par une conclusion générale qui récapitule nos contributions, identifie les limites de nos travaux et propose de nouvelles perspectives d'ouverture.

1 Informatique Sensible au Contexte

Sommaire

1.1 Introduction

Avec l'apparition de l'informatique ubiquitaire dans les deux dernières décennies, la notion de systèmes sensibles au contexte (ou Context-Aware Systems en anglais) prend de plus en plus d'importance. Les systèmes sensibles au contexte représentent une classe émergente de systèmes informatiques dont la structure et le comportement varient en fonction du contexte. Ils sont des systèmes qui s'intègrent progressivement à l'environnement physique, passent à l'arrière-plan, analysent les activités des utilisateurs, gardent une trace de ces activités et interviennent lorsque cela est nécessaire.

En effet, un système sensible au contexte consiste en un ensemble de dispositifs d'accès multiples, très hétérogènes et intégrés dans un environnement hautement dynamique. Á l'opposé d'un système traditionnel, dans lequel les dispositifs d'accès sont stationnaires, les systèmes sensibles au contexte supportent des dispositifs mobiles qui peuvent être transportés partout par les utilisateurs, afin de mieux répondre à leurs besoins. Au-delà de l'hétérogénéité, les systèmes sensibles au contexte s'inscrivent dans un environnement particulièrement dynamique et capables de percevoir le contexte de l'utilisateur et de gérer sa mobilité. Ainsi, ils utilisent les observations de contexte pour offrir des informations ou des services pertinents pour les utilisateurs.

Dans ce chapitre, nous présentons un état de l'art sur les systèmes sensibles au contexte. Mais tout d'abord, nous présentons une étude du vocabulaire de ce domaine de recherche. Nous commençons par introduire la notion de contexte en présentant les différentes définitions relatives à cette notion, ses caractéristiques essentielles et ses différents types. Ensuite, nous présentons une revue de littérature portant sur les systèmes sensibles au contexte en mettant en lumière la notion de la sensibilité au contexte. Enfin, nous décrivons les différentes fonctionnalités des systèmes sensibles au contexte.

1.2 Notion de Contexte

Au sens étymologique du terme, le mot « Contexte » prend ses origines du latin. Il est lié à deux concepts *Contextus* qui veut dire *assemblage* et *contextere* qui signifie *tisser avec*. Ainsi, l'origine latine du terme dénote l'aspect constructif et associatif de la notion de contexte. Dans cette section, nous introduisons les différentes définitions du contexte proposées dans la littérature, ses caractéristiques, et enfin, ses différents types.

1.2.1 Définitions du Contexte : Propositions de la Littérature

Dans la littérature, on trouve plusieurs définitions du mot contexte selon les disciplines et les courants de recherches qu'ils les divisent. Généralement, Le contexte peut être défini comme l'ensemble et l'assemblage des circonstances et des relations qui entourent une situation donnée.

Avant d'aborder les définitions de contexte proposées dans le domaine de l'informatique ubiquitaire, nous commençons par une définition de ce concept dans le sens littéraire, fournie par Encyclopédie Larousse : « le contexte est l'ensemble des circonstances dans lesquelles se produit un évènement, se situe une action. »

En outre, le contexte a été introduit dans différentes spécialités du domaine informatique, comme le traitement du langage naturel [Vol98, Ber99], l'aide à la décision [LM00a, JCH+04, NLW+12], la recherche d'information [XC00, BJ01, JB04] et l'informatique ubiquitaire. Nous rappelons, dans ce qui suit, des définitions tirées du domaine de l'informatique ubiquitaire. Ces définitions sont organisées selon l'ordre chronologique, démontrant une progression dans la compréhension de la notion de contexte.

Hull et al. [HNBR97] : le contexte est l'environnement d'une application.

Brown et al. [BBC97] : le contexte est les éléments de l'environnement de l'utilisateur, c'est-à-dire, l'heure, la saison, la température, l'identité et la localisation de l'utilisateur.

Ward et Hopper [WJH97] : le contexte est les états des environnements possibles de l'application.

Pascoe [Pas98] : le contexte est un sous-ensemble des états physiques et conceptuels ayant un intérêt pour une entité particulière.

Schmidt et al. [SBG99] : le contexte est la situation dans laquelle se trouve un utilisateur.

Dey [Dey01] : Le contexte est toute information utilisée pour caractériser la situation d'une entité. Une entité peut être une personne, un lieu ou un objet considéré comme pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application même [1].

Chen et Kotz [CK+00] : le contexte est l'ensemble des états environnementaux et paramètres qui déterminent le comportement d'une application ou dans lequel un événement de l'application se déroule et ayant un intérêt pour l'utilisateur [2].

Brézillon [Bré02] : le contexte est tout ce que n'intervient pas explicitement dans la résolution d'un problème mais contraint sa résolution.

1. Any information that can be used to characterize the situation of an entity. An entity is a person, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves.

2. Context is the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user.

Henricksen [Hen03] : le contexte d'une tâche est l'ensemble des circonstances qui l'entourent et qui sont considérées comme pertinentes pour sa réalisation [3].

1.2.2 Caractéristiques des Informations Contextuelles

Dans la plupart des définitions informatiques que nous avons présentées, le concept du contexte est souvent défini comme un ensemble d'informations (*informations contextuelles*) qui portent sur de nombreux éléments ayant un intérêt du point de vue de l'interaction homme-machine. Strang et Linnho-Popien [SLP03] ont définit l'information contextuelle comme toute information qui peut être utilisée pour caractériser l'état d'une entité (une personne, un lieu ou un objet) selon un aspect spécifique [4]. Ainsi, les informations contextuelles sont les informations sur lesquelles reposent principalement l'informatique ubiquitaires.

Nous soulignons ici les principales caractéristiques des informations contextuelles qui nous semblent les plus pertinentes :

— **Dynamicité** : les informations contextuelles peuvent apparaître et disparaître pendant l'exécution ou changer d'état.

— **Hétérogénéité** : les informations contextuelles issues d'un grand nombre de sources distribuées différentes sont, généralement, des informations brutes qui, sans interprétation, peuvent être dénuées de sens.

— **Incertitude** : les informations contextuelles collectées peuvent être obsolètes, incohérentes, inconsistantes ou incomplètes.

— **Interdépendance** : une information contextuelle peut être dépendante d'une autre qui est elle-même dépendante. Le changement d'une information contextuelle implique le changement de plusieurs autres.

1.2.3 Catégorisation du Contexte

Dans la littérature, plusieurs classifications ou catégorisations de contexte ont été proposées. Selon Schilit et al. [SAW94], le contexte se décompose en trois catégories où chacune des catégories répond à l'une des questions suivantes : « *Où est-tu ?* », « *Avec qui es-tu ?* », « *De quelles ressources disposes-tu à proximité ?* ». Partant de cette vision, la notion de contexte désigne tout changement qui affecte l'environnement physique, le comportement des utilisateurs et ses ressources.

Ryan et al. [RPM98] ont catégorisé le contexte en *identité* de l'utilisateur, *ressources* de

3. The context of a task is the set of circumstances surrounding it that are potentially of relevance to its completion.

4. A context information is any information which can be used to characterize the state of an entity concerning a specific aspect.

l'environnement proche, *localisation* de l'utilisateur et *période temporelle* d'exécution de l'interaction.

Dans le cadre de dispositifs mobiles, Rodden et al. [RCDD98] ont considéré différentes catégories : infrastructure, application, système, localisation et paramètres physique.

Schmidt et al. [SBG99] ont proposé de classifier les éléments de contexte selon deux catégories : *les facteurs humains* (les utilisateurs, l'environnement social, les tâches, etc.) et *l'environnement physique* (la localisation, la luminosité, la température, l'infrastructure, etc.).

Petrelli et al. [PNS+00] ont également considéré deux catégories : *le contexte social* (les aspects sociaux comme la relation entre les individus) et *le contexte matériel* (localisation, machine, plateforme existante).

Chen et Kotz [CK+00] ont catégorisé le contexte en deux classes : le *contexte actif* qui influence le comportement du système et le *contexte passif* qui est nécessaire mais pas critique pour le système.

Gwizdka [Gwi00] a élaboré une catégorisation en deux classes : le contexte *interne* contenant l'état de l'utilisateur et le contexte *externe* englobant l'état de l'environnement.

Hofer et al. [HSP+03] ont proposé deux catégories : le contexte *physique* qui peut être mesuré par les capteurs physiques et le contexte *logique* qui contient les informations sur l'interaction (l'état émotionnel de l'utilisateur, ses buts, etc.).

Kirsch et al. [KPVOGM05] ont classifié le contexte selon deux classes : le contexte *physique* décrivant les paramètres de l'utilisateur (l'emplacement, les caractéristiques du terminal et de l'application de l'utilisateur) et le contexte *organisationnel* qui réfère aux informations relatives aux processus de collaboration dans lequel l'utilisateur est impliqué (le rôle, l'activité et le calendrier).

Razzaque et al. [RDN06] ont proposé de classifier le contexte selon six catégories (voir tableau 1.1) :

— **Contexte utilisateur** : il permet d'obtenir les informations sur les utilisateurs du système informatique.

— **Contexte physique** : il offre la possibilité d'intégrer des informations relatives à l'environnement physique.

— **Contexte réseau** : il fournit des informations qui se rapportent essentiellement au réseau informatique.

— **Contexte activité** : il répertorie les évènements qui se sont déroulés dans l'environnement ainsi que leur estampille temporelle.

— **Contexte matériel** : il permet d'identifier les dispositifs de l'environnement qui peuvent être utilisés.

— **Contexte service** : il informe sur ce qui peut être obtenu.

Tableau 1.1 – Catégorisation conceptuelle [RDN06]

Catégorie	Sémantique	Exemples
Contexte utilisateur	Qui ?	Profil d'utilisateur : identifications, relations avec les autres, listes des choses à faire, etc.
Contexte physique	Où ?	Environnement physique : humidité, température, niveau de bruit, etc.
Contexte réseau	Où ?	Environnement réseau : connectivité, bande passante, protocole, etc.
Contexte activité	Ce qui se passe, quand ?	Ce qui se passe, à quel moment : entrer, sortir, etc.
Contexte matériel	Que peut-on utiliser ?	Profil et activités des dispositifs : identifications, emplacement, durée de vie de la batterie, etc.
Contexte service	Que peut-on obtenir ?	Les informations sur les fonctions qui peuvent être fournies par le système : format de fichier, affichage, etc.

Zimmermann et al. [ZLO07] ont donné une autre vision du contexte qui permet de couvrir les différents aspects pour définir le contexte. Ils ont défini le contexte comme toute information qui peut être utilisée pour caractériser la situation d'une entité. Les éléments pour la description de ces informations de contexte appartiennent à cinq catégories : individualité, activité, localisation, temps et relation. L'activité est le concept prédominant pour déterminer la pertinence des autres éléments de contexte dans une situation spécifique. Le temps et la localisation sont les concepts permettant de créer des relations entre les entités et d'autoriser l'échange d'informations contextuelles entre elles [5].

Partant des différentes classifications que nous retrouvons dans la littérature, nous proposons une catégorisation qui englobe les différents aspects de l'environnement. On considère que les informations contextuelles peuvent être classifiées selon le contexte utilisateur, le contexte informatique et le contexte environnemental (voir figure 1.2) :

— **Contexte utilisateur** : désigne une personne de la communauté ciblée par un système informatique. Il est typiquement représenté par sa capacité cognitive, sa perception, ses compétences et ses préférences.

5. Context is any information that can be used to characterize the situation of an entity. Elements for the description of this context information fall into five categories : individuality, activity, location, time, and relations. The activity predominantly determines the relevancy of context elements in specific situations, and the location and time primarily drive the creation of relations between entities and enable the exchange of context information among entities.

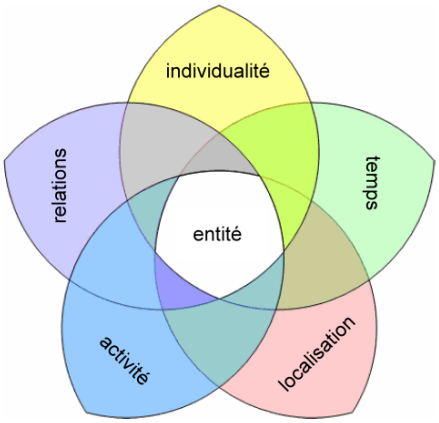


FIGURE 1.1 – Catégories fondamentales de contexte [ZLO07]

— **Contexte informatique** : désigne une plateforme de travail formée d'une structure matérielle, logicielle et réseau qui sert principalement à garantir l'interaction l'utilisateur et son environnement.

— **Contexte environnemental** : désigne l'environnement physique dans lequel une interaction peut avoir lieu. Il est représenté par un ensemble d'informations telles que : la géolocalisation, le temps, la luminosité, la température, le bruit, etc.

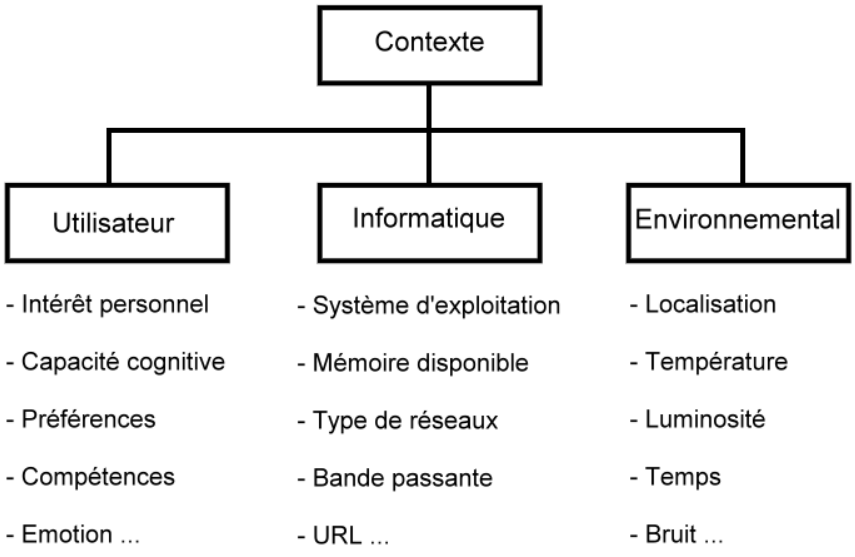


FIGURE 1.2 – Concept multidimensionnel de contexte

1.3 Systèmes Sensibles au Contexte

Après la première apparition de la notion de systèmes sensibles au contexte par Schilit et. al [SAW94], plusieurs tentatives pour définir ce type de systèmes ont été proposées [BBC97, RPM98, LS00, ADB+99]. La définition de Dey [Dey01] est la plus connue et la plus influente dans les travaux portant sur la sensibilité au contexte. Il considère que : « *un système est sensible au contexte s'il utilise le contexte pour fournir des informations et/ou des services pertinents à un utilisateur. La pertinence dépend de la tâche de l'utilisateur* » [6]. Cependant, nous remarquons dans cette définition qu'un système qui ne fait que collecter le contexte pour le fournir à une application est un système sensible au contexte légitime, même s'il ne modifie pas son comportement. Selon Rohn [Roh03], un système sensible au contexte diffère d'un système classique car il est :

— *Adaptatif* : il est capable à apprendre les préférences de l'utilisateur et à s'ajuster en conséquence.

— *Réceptif* : il anticipe les besoins de l'utilisateur dans un environnement évolutif.

— *Proactif* : il possède un objectif et est capable de prendre des initiatives, plutôt que de simplement réagir à l'environnement.

— *Autonome* : il peut agir indépendamment, sans intervention de l'utilisateur.

Xiaohang [Xia03] a constaté que la description des procédures de gestion du contexte est ignorée par la plupart des définitions. De ce fait, il a proposé une nouvelle définition qui considère qu'un système est sensible au contexte s'il peut détecter, interpréter, et exploiter des informations issues du contexte et adapter son comportement en fonction du contexte d'utilisation [7].

Cheverest et al. [CMD02] ont considéré qu'un système sensible au contexte est un système qui peut adapter la présentation des informations afin de les rendre plus pertinentes au contexte d'un utilisateur en tenant compte, par exemple, des intérêts signalés par l'utilisateur ou de son contexte environnemental. Selon Demeure et al. [DD03], un système sensible au contexte ou adaptatif au contexte est un système qui peut découvrir et utiliser des informations contextuelles telles que la localisation de l'utilisateur, les caractéristiques de ses dispositifs, etc.

Abbas et al., 2007 [AVF07] ont défini les systèmes sensibles au contexte comme des systèmes qui sont capables de retourner à l'utilisateur des informations pertinentes et adaptées à ses besoins et son contexte qui influence son comportement lors de son interaction avec les systèmes d'information.

6. A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

7. Context-awareness is the ability for a software system to acquire, manage, interpret, and response to context to provide appropriate services to the changing situation.

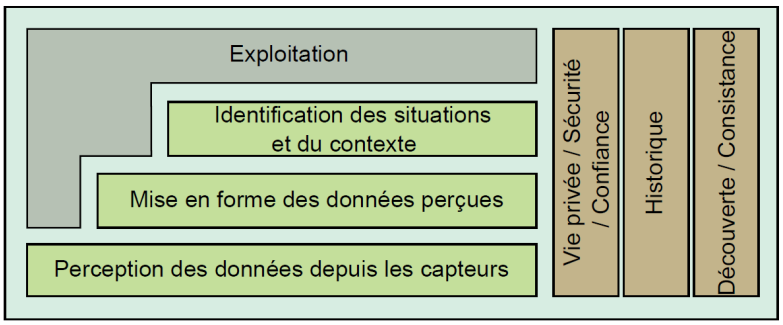


FIGURE 1.3 – Modèle conceptuelle de systèmes sensibles au contexte [CCDG05]

Coutaz et al. [CCDG05] ont proposé un modèle général (voir figure 1.3) pour les systèmes sensibles au contexte, illustrant les différents niveaux de son infrastructure, de la récupération des données contextuelles à leur exploitation.

Baldauf et. al [BDR07] ont fait un état de l'art sur les systèmes sensibles au contexte, où ils ont extrait les traits communs de tels systèmes dans la pratique. Les auteurs ont proposé une plateforme conceptuelle à différentes couches pour la séparation des préoccupations dans les systèmes sensibles au contexte. Les couches que les auteurs considèrent sont présentées dans la figure 1.4. Selon ces auteurs, les systèmes sensibles au contexte se caractérisent par leur capacité à adapter leur fonctionnement, par la prise en compte du contexte environnant, afin d'augmenter leur utilisabilité et leur efficacité.

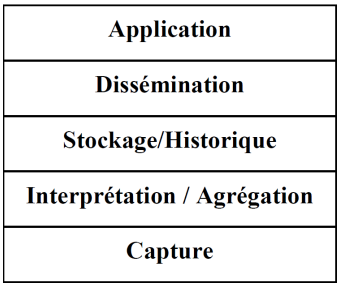


FIGURE 1.4 – Plateforme conceptuelle de systèmes sensibles au contexte [BDR07]

En se basant sur les différentes définitions des systèmes sensibles au contexte, nous pouvons dire d'un système qu'il est sensible au contexte s'il est caractérisé par des fonctionnalités qui lui permettent d'*acquérir* des informations de contexte en interagissant avec des capteurs (acquisition du contexte), d'*interpréter* ces informations et de les analyser pour détecter les changements pertinents qu'ils subissent, de *stocker* les informations capturées et interprétées pour une utilisation ultérieure (représentation du contexte), et enfin, de *transmettre* les différentes informations aux applications pour les *adapter* au contexte, lors de la détection des situations pertinentes (adaptation au contexte).

Dans ce qui suit, nous présentons tout d'abord la notion de la sensibilité au contexte. Ensuite, nous décrivons les différentes fonctionnalités des systèmes sensibles au contexte. Nous commençons par la présentation des différentes techniques utilisées permettant d'acquérir les informations contextuelles. Puis, nous mettons en évidence l'importance de sa représentation en citant les différents modèles proposés pour représenter le contexte. Enfin, nous présentons la fonctionnalité d'adaptation au contexte et ses différents types.

1.3.1 Sensibilité au Contexte

La sensibilité au contexte (*Context-Awareness*) est la propriété principale d'un système ubiquitaire caractérisant sa capacité de réagir proprement en prenant en compte l'information de contexte. Schilit et al. [SAW94] ont été les premiers à introduire la notion de la sensibilité au contexte, à travers leur système de localisation « *Active Map Service* ». Il ont défini la sensibilité au contexte comme la capacité d'une application d'un utilisateur mobile à s'adapter aux changements survenant dans l'environnement où ils sont situés. Ils ont donc considéré la sensibilité au contexte comme la sensibilité à la situation dans laquelle se trouve un utilisateur.

Une autre définition est ensuite proposée par Brown [Bro95]. Il a défini la sensibilité au contexte dans son travail relatif à un guide touristique « *The Stick-e Document* » comme toute application qui prend en compte le contexte de l'utilisateur.

Pour Hull et al. [HNBR97] ainsi que Pascoe et al. [PRM98], la sensibilité au contexte est la capacité d'un système à détecter, interpréter et réagir aux changements de l'environnement local de l'utilisateur et des dispositifs qu'il utilise.

Salber et al. [SDA98] ont défini la sensibilité au contexte comme étant la capacité d'un système à agir en temps réel avec des données provenant du contexte.

Ryan et al. [RPM98] ont défini la sensibilité au contexte en tant que la capacité d'un système à percevoir et à agir sur des informations sur son environnement, telles que la localisation, le temps, la température ou l'identité de l'utilisateur.

Selon Abowd et al. [ADB+99] une application est sensible au contexte si elle est capable de percevoir son contexte et si elle possède un comportement dynamique guidé par sa connaissance sur son environnement.

Lieberman et Selker [LS00] ont considéré que, dans une application sensible au contexte, le système tient compte des informations qui ne lui sont pas fournies explicitement (autrement dit, des informations qu'il perçoit ou récupère par lui-même, indépendamment de l'utilisateur). Dans une telle vision, toute information n'appartenant pas aux entrées ou aux sorties explicites d'une application mais influençant néanmoins son traitement appartient

alors à son contexte.

Korkea-Aho [KA00] a donné une définition qui considère que la sensibilité au contexte d'un système est la capacité à percevoir, interpréter et utiliser les différentes informations relatives au contexte courant pour adapter dynamiquement sa fonctionnalité.

Henricksen [Hen03] a considéré qu'une application est sensible au contexte si elle s'adapte aux changements de l'environnement et aux besoins de l'utilisateur.

Barkhuus [Bar03] a donné une définition proche de celle énoncée par Schilit et al. [SAW94], en caractérisant la sensibilité au contexte comme la capacité des applications à détecter et réagir aux variables de l'environnement de façon autonome.

Selon Xiaohang [Xia03], la sensibilité au contexte d'un système logiciel est sa capacité à acquérir, gérer, interpréter et répondre aux changements du contexte afin de fournir les services appropriés.

Partant des différentes définitions et compréhensions de la sensibilité au contexte, nous pouvons souligner deux aspects principaux qui caractérisent ce concept :

— **Perception** : la capacité à percevoir et détecter son contexte ; cette notion est présente dans toutes les définitions.

— **Réaction** : la capacité d'agir différemment en fonction du contexte perçu.

1.3.2 Acquisition du Contexte

L'acquisition du contexte est une tâche principale lors de la conception d'un système sensible au contexte. Elle repose sur la capacité d'un système à capturer des informations de l'environnement jugées pertinentes et à détecter les changements effectués. L'acquisition des informations de contexte peut se faire selon plusieurs méthodes, indépendamment de l'application. Indulska et Sutton [IS03] ont considéré que le capture des informations de contexte dépend de la source du contexte et du type des informations collectées. Pour cela, ils ont défini trois types de capteurs :

— **Capteur physique** : représente un dispositif matériel qui peut être intégré ou non dans d'autres appareils et permettant de capturer des informations de l'environnement autour de l'utilisateur et du système lui-même, telles que la température, la localisation géographique, niveau de bruit, lumière, etc. Par exemple, l'utilisation de GPS (Global Positioning System) pour déterminer les coordonnées géographiques d'un utilisateur, ou encore, l'utilisation de capteurs de mouvement pour allumer automatiquement la lumière. Le tableau 1.2 présente quelques exemples de capteurs physiques selon le type d'information qu'ils fournissent.

— **Capteur virtuel** : correspond à une application ou à un service logiciel capable de fournir des informations sur le contexte. Par exemple, la tâche actuelle d'un employé peut être extraite à partir de son planning électronique. Les capteurs virtuels sont généralement moins coûteux que les capteurs physiques puisqu'ils sont basés sur des composants logiciels qui sont moins chers que des appareils électroniques.

— **Capteur logique** : utilise les informations des capteurs physiques et virtuels pour déduire des informations de haut niveau, telles que les caractéristiques sociales, économiques, les compétences, etc. [Cha07]. Par exemple, le système peut analyser l'historique d'un utilisateur afin de mettre à jour ses préférences.

Tableau 1.2 – Exemples de capteurs de contexte [SVL01]

Type d'information	Capteurs disponibles
Lumière	Photodiodes, capteurs de couleurs, capteurs d'infrarouge (IR) et d'ultra violet (UV)
Contexte visuel	Caméras numériques
Audio	Microphones
Mouvements et accélérations	Interrupteurs, capteurs d'angles, accéléromètres, détecteurs de mouvement, champs magnétique
Localisation	Extérieur : GPS (Global Positionning System), GSM (Global System for Mobile Communications). Intérieur : Active Badge System
Touche	Capteurs tactiles
Température	Thermomètres numériques
Caractéristiques biologiques	Capteurs Biométriques (résistance de peau, tension)

De même, Mostefaoui et al. [MPRB04] ont classé ces méthodes en trois catégories :

— **Acquisition par capteurs** : consiste à utiliser des capteurs physiques ou virtuels pour récupérer des informations de contexte, telles que la température, la pression, la lumière, la bande passante, la charge de la batterie, etc. L'information de contexte collectée à l'aide de captures est dynamique, il est donc nécessaire de mettre à jour les observations fréquemment.

— **Acquisition par dérivation** : consiste à utiliser un ou plusieurs informations de contexte pour déduire ou calculer à la volée une information de plus haut niveau en utilisant des méthodes d'interprétation ou de raisonnement. Par exemple, l'heure et la date.

— **Acquisition par profil** : consiste à récupérer des informations soit à travers une interface graphique soit par l'intermédiaire d'un fichier de profil. Ces informations sont fournies explicitement telles que le profil utilisateur, les informations sur une machine connectée au réseau, etc.

Selon Chen [Che04], l'accès des applications au contexte peut se faire de trois manières :

— **Accès direct** : dans ce cas, l'application s'interface directement avec la source du contexte. Pour ce faire, l'application intègre les pilotes logiciels et les APIs de communication avec les capteurs, ce qui rend cette méthode peu adaptée pour les systèmes distribués. De plus, elle manque de scalabilité à cause du fort couplage entre les applications et les pilotes.

— **Infrastructure intergicielle** : cette méthode est plus aisément extensible que la précédente. Elle consiste à utiliser une infrastructure intergicielle pour séparer le code métier des détails de bas niveau concernant la capture du contexte. Les infrastructures intergicielles agissent comme un intermédiaire entre l'application et les capteurs, permettant à l'application d'acquérir les informations de contexte.

— **Serveurs de contexte** : contrairement aux deux premières techniques, cette dernière consiste à externaliser la gestion du contexte en utilisant une architecture client-serveur. Un serveur possède plus de ressources qu'un terminal mobile, et pourra donc mieux gérer une grande quantité d'information de contexte, facilitant ainsi la tâche du client. Le serveur de contexte centralise les informations fournies par les capteurs, et permet aux clients (applications) d'acquérir ces informations soit par interrogation ou par notification. Le mode d'acquisition par interrogation consiste à envoyer une requête vers le serveur pour récupérer l'état du contexte, alors que le mode d'acquisition par notification consiste à s'abonner auprès du serveur pour être notifiés au moment d'un changement de contexte.

1.3.3 Représentation du Contexte

La gestion de contexte et de ses différentes caractéristiques nécessite de représenter le contexte explicitement dans le système informatique. La représentation du contexte consiste à définir une syntaxe, ou plus généralement, une structure concrète permettant d'organiser les informations de contexte capturées. Les informations de contexte doivent donc être construites de manière structurée et ordonnée pour qu'elles soient exploitées de façon efficace et facile. Selon Brézillon [Bré02] « une représentation efficace du contexte en machine, tant en termes de modélisation de connaissances que de raisonnement à partir de celles-ci, est un problème à résoudre, et ce, aussi bien du point de vue de la programmation que de son utilisation ».

Par conséquent, la modélisation du contexte est une étape indispensable dans le processus de développement de systèmes sensibles au contexte. Elle consiste en la définition de concepts, de relations entre ces concepts et des contraintes sur ces concepts. En d'autres termes, il est impératif de définir un modèle de contexte structurellement abstrait, et sémantiquement riche pour l'interprétation et le stockage des informations de contexte. Selon Henricksen [Hen03], « un modèle de contexte identifie un sous-ensemble du contexte qui est accessible

de façon réaliste à travers des capteurs, des applications et des utilisateurs et qui peut être exploité pour l'exécution de la tâche. Le modèle de contexte qui est utilisé par une application sensible au contexte est généralement spécifié par le développeur de l'application, mais peut changer dans le temps ».

La diversité des informations de contexte et leur utilisation dans divers domaines applicatifs engendrent différentes façons de les modéliser. Ainsi, le choix du modèle de contexte joue le rôle centrale dans cette étape. Plusieurs approches de représentation et de modélisation du contexte ont émergé dans la littérature et ont été classifiées par plusieurs travaux tels que Strang et Linnhoff-Popien [SLP04], Chaari et al. [CLF05] et [BBH+10]. Un aperçu de ces approches est présenté ci-dessous.

— **Approches basées sur les algèbres de processus**
  - Calcul des Systèmes Communicants : CCS [Mil80]
  - π-calcul [MPW92]
  - Ambiants mobiles [CG00]
  - Calcul d'Action [Mil96]

— **Approches basées sur les paires clé-valeur**
  - Capeus [SMLP02]
  - Amalthaea [Mou97]
  - Fab [BS97]
  - Letizia [L+95]

— **Approches basées sur les langages de balisage**
  - ConteXtML [Rya99]
  - Comprehensive Structured Context Profiles (CSCP) [BHH04]
  - Centaurus Capability Markup Language (CCML) [KKC+01]

— **Approches graphiques**
  - Context Composition graph (CoCo) [BKLPS04]
  - Context UnifiedModeling Language (ContextUML) [SB05]
  - ContextModeling Language (CML) [HI06]
  - Modeling Language for Context-Aware Systems (MLContext) [HPGMB11]

— **Approches orientées objet**
  - Système GUIDE [CMD02]
  - HYDROGEN [HSP+03]

— **Approches basées sur les ontologies**
  - Context Broker Architecture Ontology (COBRA-ONT) [CFJ03, CFJ04b, CFJ04a]
  - Context Ontology (CONON) [GWPZ04, WZGP04]
  - Context Ontology Language (CoOL) [SLPF03]

— **Approches basées sur les systèmes réactifs bigraphiques**

Modèle platographique [BDE+06]

Modèle de contexte et capacités [XXL11]

Modèle de contexte et actions [WXL11]

Modèle BiAgents [PKS12]

Nous détaillons dans le chapitre 2 les approches de modélisation des systèmes sensibles au contexte les plus répondues dans la littérature.

1.3.4 Adaptation au Contexte

L'adaptation d'un système au contexte de l'utilisateur est devenue un des enjeux majeurs des vingt dernières années. Selon le grand dictionnaire, l'adaptation est définie comme « l'opération qui consiste à apporter des modifications à un logiciel ou à un système informatique dans le but d'assurer ses fonctions et, si possible, d'améliorer ses performances, dans un environnement d'utilisation ».

Généralement, en informatique, l'adaptation peut prendre deux formes : adaptabilité et auto-adaptabilité. D'après l'encyclopédie Larousse, l'auto-adaptabilité signifie « l'aptitude d'un système à modifier ses paramètres de structure de manière à ce que son fonctionnement demeure satisfaisant en dépit des variations de son environnement ». Cependant, l'adaptabilité est définie comme étant « la capacité de s'adapter à de nouveaux milieux ou à de nouvelles situations ».

En informatique ubiquitaire, plusieurs définitions ont été proposées par les chercheurs qui se sont penchés sur ce concept. D'après Riveill et al. [RPPM00], l'adaptation est la capacité d'harmoniser l'application avec son environnement. Ainsi, l'exécution de cette application doit nécessairement prendre en compte non seulement les besoins de l'utilisateur mais aussi le contexte pour garantir l'accès aux services et aux informations pertinentes.

Subramanian et al. [SC01], ont considéré une adaptation au contexte comme un changement dans le système pour prendre en compte un changement dans l'environnement du système.

Selon David [Dav05], « une adaptation est une modification d'un système, en réponse à un changement dans son contexte, avec l'objectif que le système résultant soit mieux à même pour réaliser sa fonction dans le nouveau contexte ».

Chassot et al. [CGD+06] ont défini deux types d'adaptation qui dépendent de la manière de son application : statique ou dynamique.

— **Adaptation statique** : correspond à une adaptation effectuée avant l'exécution de l'ap-

plication, en fonction des connaissances détenues de l'environnement de déploiement, ou à une adaptation faite pendant une phase d'initialisation de l'application.

— **Adaptation dynamique** : correspond à une adaptation effectuée pendant l'exécution de l'application, en fonction de l'environnement de déploiement.

Selon Ketfi et al. [KBC02], les raisons conduisent à adapter une application peuvent être classées en quatre catégories :

— **Adaptation correctionnelle** : consiste à identifier le composant de l'application défectueux et à le remplacer par un autre. Le nouveau composant fournit la même fonctionnalité que l'ancien, il se contente simplement de corriger ses défauts.

— **Adaptation adaptative** : consiste à adapter l'application en réponse aux changements affectant son environnement d'exécution, même si elle s'exécute correctement.

— **Adaptation évolutive** : consiste à étendre l'application en ajoutant des nouvelles fonctionnalités pour satisfaire les besoins évolutifs de l'utilisateur.

— **Adaptation perfective** : désigne les améliorations de performances apportées à l'application.

Raibulet [Rai08] a classifié l'adaptation en six catégories :

— **Adaptation architecturale** : met l'accent sur les changements apportés, en temps réel, à la structure du système et/ou aux interactions entre ses composants en utilisant un modèle architectural du système [GCH+04].

— **Adaptation compositionnelle** : concerne les modifications de la structure et du comportement du système, en temps réel, en réponse aux changements apportés à son environnement d'exécution [MSKC04]. Par exemple, le changement de la signature d'une méthode.

— **Adaptation structurelle** : consiste à mettre à jour la structure du système en préservant son comportement et ses services [BSO07].

— **Adaptation comportementale** : désigne les changements dynamiques apportés lors de l'exécution des composants du système de manière non intrusive [GLT08].

— **Adaptation de contenu** : consiste en la transformation et la manipulation des contenus en se basant sur les caractéristiques propres à l'application et au terminal utilisés [HGH+07].

— **Adaptation de service** : se concentre sur la sélection dynamique de services, selon leurs disponibilités, en réponse à la requête de l'utilisateur [CY07].

1.4 Conclusion

Les systèmes sensibles au contexte représentent une nouvelle génération de systèmes informatiques dans laquelle les technologies sont graduellement intégrées à l'environnement

physique. Ils sont des systèmes capables de percevoir le contexte de l'environnement qui les entoure afin d'adapter en conséquence leur comportement en fonction des besoins de l'utilisateur. Ainsi, nous disons d'un système qu'il est sensible au contexte s'il peut acquérir, interpréter et exploiter des informations issues du contexte et adapter sa réponse en fonction du contexte d'utilisation.

Dans ce chapitre, nous avons présenté un état de l'art sur l'informatique sensible au contexte. Au début de ce chapitre, nous avons abordé les différentes définitions de la notion de contexte en remontant à l'origine du terme dans le langage courant, puis dans le domaine informatique. Ensuite, nous avons présenté les principales caractéristiques des informations contextuelles. Puis, nous avons étudié ses différentes catégorisations proposées dans la littérature, et proposé notre propre catégorisation. Nous avons présenté, par la suite, une revue de littérature portant sur la notion de la sensibilité au contexte en montrant l'importance de cette notion dans le domaine d'informatique moderne. Enfin, nous avons présenté une étude bibliographique sur les systèmes sensibles au contexte présentant leurs caractéristiques et leurs différentes fonctionnalités.

Dans le chapitre suivant nous présentons un état de l'art sur les approches de modélisation des systèmes sensibles au contexte.

2 Approches de Modélisation des Systèmes Sensibles au Contexte

Sommaire

2.1 Introduction

Pour décrire la sensibilité au contexte d'un système informatique, il faut déterminer l'ensemble de contextes auquel ce système est sensible et le décrire dans un modèle. Par conséquent, la sensibilité au contexte est un aspect central de la modélisation. Le contexte est donc très important, car il offre des informations détaillées sur les entités des systèmes, telles que les personnes, les lieux et les objets. Chaque entité est caractérisée par une identité qui se réfère à la possibilité d'avoir un identifiant unique, la localisation qui représente l'information spatiale de l'entité, le statut (ou l'activité) qui représente leurs propriétés intrinsèques. Enfin, le temps qui se réfère à l'ordonnancement d'événements. En outre, le contexte change toujours entraînant des modifications du comportement des systèmes sensibles au contexte. Ces derniers peuvent être en mesure de se contrôler et de s'adapter en réponse aux changements de contexte. Cependant, l'hétérogénéité et la diversité des informations contextuelles et leur intérêt dans divers domaines engendrent différentes façons de les modéliser.

Dans ce chapitre, nous classifions les approches de modélisation des systèmes sensibles au contexte en quatre catégories : les approches orientées modèles, les approches orientée ontologies, les approches orientées algèbres de processus, et les approches orientées systèmes réactifs bigraphiques. Cette catégorisation se base principalement sur le principe sur lequel repose la modélisation, l'expressivité du modèle et la possibilité de le réutiliser. Puis, nous faisons une comparaison entre ces différentes approches afin de pouvoir aboutir au choix final du formalisme qui sera le mieux adapté à la modélisation des systèmes sensibles au contexte.

2.2 Approches Orientées Modèles

L'objectif principal des approches orientées modèles est d'élever le niveau d'abstraction en passant du niveau code au niveau modèle du développement. En effet, ces approches sont basées sur un modèle adapté à l'encapsulation et la réutilisation des informations contextuelles, la vérification rapide et simple des propriétés attendues de l'application, et la génération automatique du code de l'application à partir de son modèle. Nous rappelons, dans ce qui suit, quelques travaux développés dans la littérature au sujet de l'utilisation de ces approches, leurs concepts fondamentaux et les principales propositions de leur mise en œuvre.

2.2.1 Langage ContextUML

ContextUML (*Context Unified Modeling Language)* [SB05] est un méta-modèle basé sur une extension du modèle UML (*Unified Modeling Languag*e) qui permet de modéliser les services web sensibles au contexte. La syntaxe de ContextUML comprend un méta-modèle et une notation. Le méta-modèle présenté dans la figure 2.1 est composé de plusieurs classes

définissant la syntaxe abstraite du langage ContextUML, alors que la notation définit le format concret utilisé pour représenter le langage (aussi appelé syntaxe concrète).

— La classe **Context** permet de décrire un contexte observable qui peut être soit un contexte de bas niveau AtomicContext ou bien un contexte de haut niveau CompositeContext. La classe AtomicContext représente les contextes qui ne dépendent pas d'autres contextes et qui sont fournis directement par les sources de contexte. Alors que, la classe CompositeContext représente les contextes composites qui regroupent de multiples contextes, soit atomiques ou composites. La notion de contexte composite peut être utilisée pour fournir un vocabulaire de modélisation riche.

— La classe **ContextSource** représente les ressources à partir de lesquelles les contextes sont récupérés. Ces ressources peuvent être simples (ContextService) ou composées (ContextServiceCommunity). Une instance ContextService consiste en un fournisseur autonome permettant la collection, le raffinage et la diffusion des informations contextuelles. Une instance ContextServiceCommunity regroupe un ensemble des instances de ContextService offrant une interface unifiée. Elle est conçue comme un support de récupération des informations contextuelles à partir des ressources hétérogènes et dynamiques. La classe ContextServiceCommunity permet une sélection de service appropriée selon certains critères de Qualité de Contexte (QoC) [BKS03] tels que la précision de l'information contextuelle (precision), la probabilité d'exactitude de cette information (correctnessProbability), et la mise à jour du contexte (refreshRate).

— La classe **CAMechanism** est une classe qui formalise les mécanismes de la sensibilité au contexte. Ces mécanismes sont affectés à des instances de la classe CAObjects appelés objets sensibles au contexte. La classe CAObject est une classe de base de tous les éléments du modèle ContextUML qui représentent des objets sensibles au contexte. Il existe quatre sous types de la classe *CAObject* : Service,Operation, Message, et Part. Chaque service offre une ou plusieurs opérations et chaque opération appartient à un service précis. Chaque opération peut avoir un ou plusieurs messages d'entrée/sortie. De même, chaque message peut avoir plusieurs paramètres qui peuvent d'être des instances de la classe Part. En outre, la classe ContextServiceCommunity est catégorisée selon deux sous types : ContextBinding et ContextTreggering. La classe ContextBinding modélise le mapping entre les instances de la classe Context et les objets sensibles au contexte de la classe CAObject. Ce mapping permet de récupérer automatiquement des informations pour les utilisateurs en fonction de l'information de contexte disponible. La classe ContextTreggering modélise l'adaptation contextuelle où les services peuvent être automatiquement exécutés ou modifiés en fonction des informations de contexte. Elle est composée des ensembles de contraintes contextuelles et des actions où une action ne peut être exécutée si et seulement si toutes les contraintes contextuelles correspondent à cette action sont vraies.

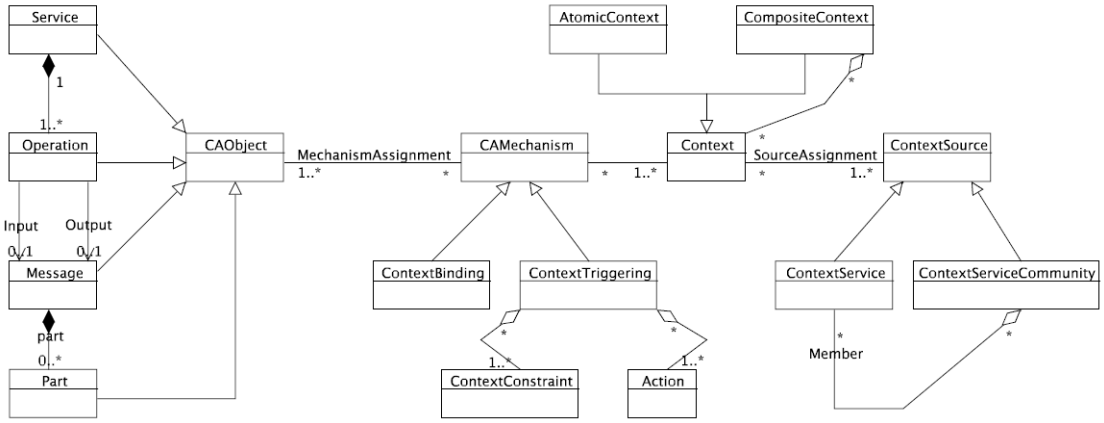


FIGURE 2.1 – Méta-modèle de ContextUML [SB05]

2.2.2 Langage CML

CML (*Context Modeling Language*) [HI06] est un langage orienté objet dérivé de l'ORM (Object-Role Modeling) [Hal09] permettant la spécification formelle des informations de contexte et leurs propriétés telles que sa qualité et sa validité temporelle. Il fournit des structures de modélisation pour la description des différents types d'informations, leurs classifications : information capturée (Sensed), information statique (Static), information de profil (Profiled), ou information dérivée(Derived), et les dépendances entre ces informations.

Dans le modèle présenté par la figure 2.2, les informations de contexte sont regroupées en un ensemble d'entités. Chaque entité modélise un objet conceptuel ou physique tels qu'une personne (Person), dispositif (Device), ou un canal de communication (Communication Channel) représentés par des ellipses. Les propriétés des entités sont représentées par des attributs (name, idenity). Les entités sont liées entre eux à travers des relations d'associations (has channel, has mode, located at, etc...) où chaque association est typée par l'information qu'elle prote ($\wedge\wedge, s, \circ, *$). Enfin, CML prend en charge à la fois les contraintes générales telles que la cardinalité des relations ($\leftrightarrow, a$), et les contraintes spécifiques telles que les contraintes de temps (=, []).

La figure 2.2 représente un exemple de modélisation CML d'une application de communication sensible au contexte. La modélisation couvre les activités des utilisateurs dans une forme d'un type de fait temporel qui indique les activités passées, présentes et futures, les associations entre ces utilisateurs, les canaux de communications et les terminaux et les emplacements de ces utilisateurs et de leurs terminaux.

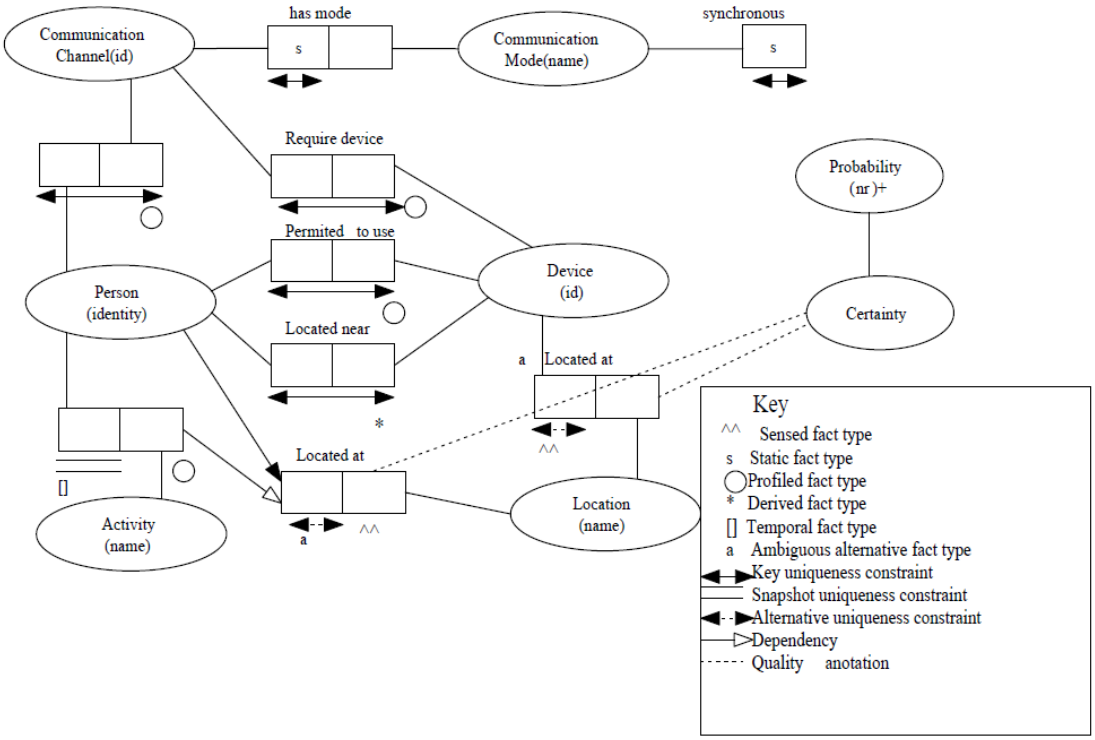


FIGURE 2.2 – Exemple d'un modèle CML [HPGMB11]

2.2.3 Langage MLContext

MLContext (*Modeling Language for Context-Aware Systems*) [HPGMB11] est un langage dédié (Domain Specific Language) textuel conçu pour modéliser les informations de contexte et pour générer automatiquement des artéfacts logiciels à partir des modèles de contexte. Les trois principaux concepts du langage MLContext sont : l'entité, le contexte et la source du contexte, où un contexte peut également être composé d'autres contextes connexes. Contrairement à d'autres approches, la modélisation en MLContext est centrée sur les *entités* plutôt que les catégories, c'est-à-dire, les instances des entités sont les éléments clés de la spécification MLContext parce que les sources de contexte ne sont pas liées aux catégories mais plutôt aux entités. En fait, deux entités de la même catégorie peuvent avoir des sources de contexte différentes pour la même propriété. Une autre raison est que le développeur doit fournir des informations détaillées sur chaque entité lorsque des instances d'une catégorie sont créées. MLContext peut déduire la catégorie en fonction des propriétés des entités qui lui appartiennent. Les changements dans les propriétés des entités seront automatiquement répercutés sur leurs catégories.

En outre, MLContext permet de spécifier une entité générique : une entité sans valeurs spécifiques de ses attributs, à partir de laquelle la catégorie est générée. Les types de contextes

ont été organisés dans une taxonomie de contexte qui comprend les types les plus utilisés : *physique*, *environnement*, *calcul*, *personnel*, *social* et *tâche*. Le contexte d'une entité est généralement composé de plusieurs contextes de différents types, et une distinction peut donc être faite entre le contexte complexe (le contexte *parent*) et les contextes simples (les contextes *fils*) dont il est composé. Les contextes fils sont constitués d'un ensemble d'informations contextuelles de même type qui peuvent provenir de différentes sources.

En effet, MLContext a été conçu pour fournir :

— **Abstraction** : Le langage permet une abstraction de haut niveau par le biais de constructions qui sont proches des concepts de domaine (par exemple entité, le contexte ou type du contexte) pour encourager les utilisateurs qui ne sont pas des développeurs à participer à la modélisation de contexte.

— **Plateforme indépendante** : Le langage permet la création des modèles de contexte sans se préoccuper des détails d'implémentation.

— **Domaine d'application indépendant** : Puisque les contextes ont beaucoup de similitudes dans les différentes applications sensibles au contexte, MLContext permet la modélisation des contextes indépendamment du domaine sensible au contexte.

— **Typage** : Le langage offre la possibilité de modéliser les différents types de contexte, tels que le contexte physique, social ou de calcul. En outre, la séparation des informations en plusieurs contextes peut être exploitée pour réaliser des améliorations telles que des méthodes de stockage et de récupération plus efficaces.

— **Mécanismes de traçabilité, qualité et temps** : Le langage fournit un mécanisme de traçabilité pour soutenir les informations historiques, un mécanisme pour mesurer la qualité des informations fournies, et il supporte à la fois les informations temporelles statiques et dynamiques.

— **Réutilisation** : Le langage permet la réutilisation des modèles, ce qui signifie que les modèles de contexte peuvent être réutilisés dans différentes applications en utilisant le même contexte.

— **Utilisabilité** : Enfin, le langage est facile et intuitif à utiliser.

Le méta-modèle de MLContext est représenté sur la figure 2.3. Comme expliqué précédemment, MLContext a été conçu autour de la notion d'entité (*Entity metaclass*), et toutes les informations contextuelles se réfèrent toujours à une entité. Un contexte complexe (*ComplexContext*) est une agrégation des contextes simples (*SimpleContext*) de différents types (par exemple, *physique* et *social*). Par exemple, le contexte d'une personne peut être une agrégation d'un contexte personnel (*nom* et *âge*) et un contexte physique (*température*). Un contexte simple est constitué d'un ou plusieurs éléments d'information de contexte (*ContextInformation metaclass*) de même type (par exemple, le nom et l'âge sont des informations de contexte personnels). Les informations d'un contexte simple peuvent contenir une référence à d'autres

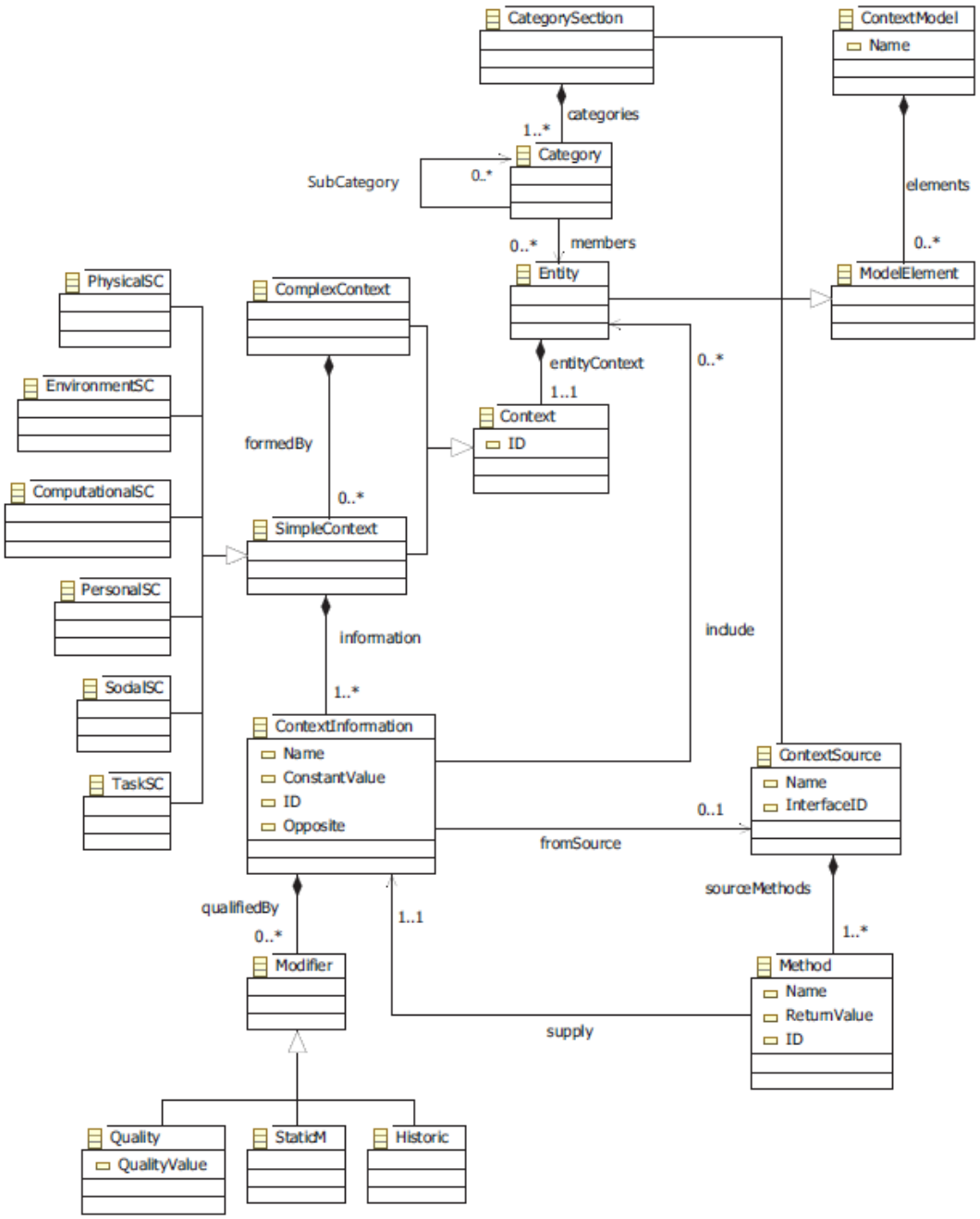


FIGURE 2.3 – Méta-modèle de MLContext [HPGMB11]

entités du modèle (*include*). La valeur d'une information de contexte (*ContextInformation*) peut être une constante ou peut provenir d'une source contexte (*ContextSource*) comme l'indique la relation *fromSource*. L'information provenant de la source est fournie par une méthode (*Method*) de l'interface du dispositif physique. La même source peut fournir des informations de contexte à plusieurs entités et peut utiliser une méthode différente pour

chacune d'entre elles. La hiérarchie de modificateur (*Modifier*) représente les trois modificateurs optionnels pour les informations de contexte : la qualité (*Quality*) pour spécifier l'exactitude de l'information, historique (*Historic*) pour garder la trace de l'information, et statique (*StaticM*) pour spécifier que l'information ne change pas avec le temps. La méta-classe *CategorySection* représente l'ensemble des catégories du modèle. Chaque catégorie (*Category*) contient toutes les entités appartenant à cette catégorie, et peut également être une sous-catégorie. Enfin, une entité ou une sous-catégorie peuvent appartenir à plus d'une catégorie (c'est-à-dire classification multiple).

2.3 Approches Orientées Ontologies

Une ontologie est une structure sémantique et formelle dans laquelle les concepts d'un domaine et leurs inter-relations sont définis. Dans la littérature, la définition la plus citée est celle donnée par [Gru93] :

"Une ontologie est une spécification explicite d'une conceptualisation. Il s'agit de représenter un domaine de connaissance dans un formalisme déclaratif. L'ensemble d'objets qui peuvent être représentés sont appelés univers du discours".

En effet, plusieurs approches basées sur les ontologies ont été proposées pour décrire le contexte et ses propriétés. Ces approches permettent non seulement de modéliser le contexte, mais aussi de raisonner sur celui-ci. Nous présentons, dans ce qui suit, trois ontologies dédiées au contexte : COBRA-ONT (Context Broker Architecture Ontology), CONON (CONtext ONtology), et CoOL (Context Ontology Language) dans le but d'étudier les différentes représentations ontologiques des informations de contexte dans chaque approche, le type de liens entre ces informations et le domaine d'utilisation de chaque ontologie.

2.3.1 Ontologie COBRA-ONT

COBRA-ONT (*Context Broker Architecture Ontology*) [CFJ03, CFJ04b] est une ontologie écrite en OWL-DL (*Ontology Web Language Description Logics*) [CFJ05] pour étendre l'architecture COBRA (Context Broker Architecture) [CFJ04a]. L'objectif principal de COBRA-ONT est la modélisation des contextes d'un système de salle de réunion intelligent dans le but de les partager entre ses agents. Ce partage offre au système la possibilité de raisonner sur les informations contextuelles collectées et de gérer les incohérences des données qui peuvent apparaître.

En effet, trois versions de COBRA-ONT ont été développées. La première version est COBRA-ONT (v0.2) [CFJ03] qui vise à modéliser les *lieux*, les *agents* et les *activités*. Quatre sous-

ontologies ont été proposées permettant de décrire des informations de localisation en général, des informations d'agents, la localisation de ces agents et leurs activités. Ensuite, la version COBRA-ONT (v0.3) [CFJ04b] comprend une ontologie de localisation (*Space*), une ontologie de dispositif (*Device*), et une ontologie temporelle (*Time*). L'ontologie de localisation physique comprend des frontières géographiques (pièces, bâtiments, etc.), des propriétés spatiales (lieux atomiques, lieux composés, etc.), et des propriétés temporelles (salles de réunion pendant les heures de travail, les bureaux pendant un jour férié, etc.). Enfin, la version COBRA-ONT(v0.4), comme le montre la figure 2.4, est étendue avec quatre nouvelles sous-ontologies : *Actions*, *Agent*, *Meeting*, et *Document*.

En résumé, les ontologies proposées par COBRA-ONT sont les suivantes :

— **Action** : une ontologie qui représente l'ensemble de toutes les actions effectuées par un agent.

— **Agent** : une ontologie conçue en partant du modèle BDI (croyance, désir, intention) des agents intelligents. un agent peut être une personne ou un agent logiciel.

— **Device** : une ontologie pour décrire les profils de différents dispositifs physiques. Chaque dispositif a un profil unique défini par certaines caractéristiques.

— **Document** : une ontologie placée sur un serveur web en tant que document web, qui peut être téléchargé ou référencé par d'autres ontologies.

— **Meeting** : une ontologie pour la description des informations associées aux réunions, calendriers d'événements, et participants d'un événement. Elle peut aider les systèmes de réunion intelligents de représenter et raisonner sur le contexte d'une réunion.

— **Space** : est une ontologie conçue pour supporter le raisonnement sur les relations spatiales entre les différents types de régions géographiques, le mapping des coordonnées géospatiales vers une représentation symbolique de l'espace et vice versa, et la représentation des mesures géographiques de l'espace.

— **Time** : est une ontologie conçue pour exprimer le temps et les propriétés temporelles des différents événements qui se produisent dans le monde physique.

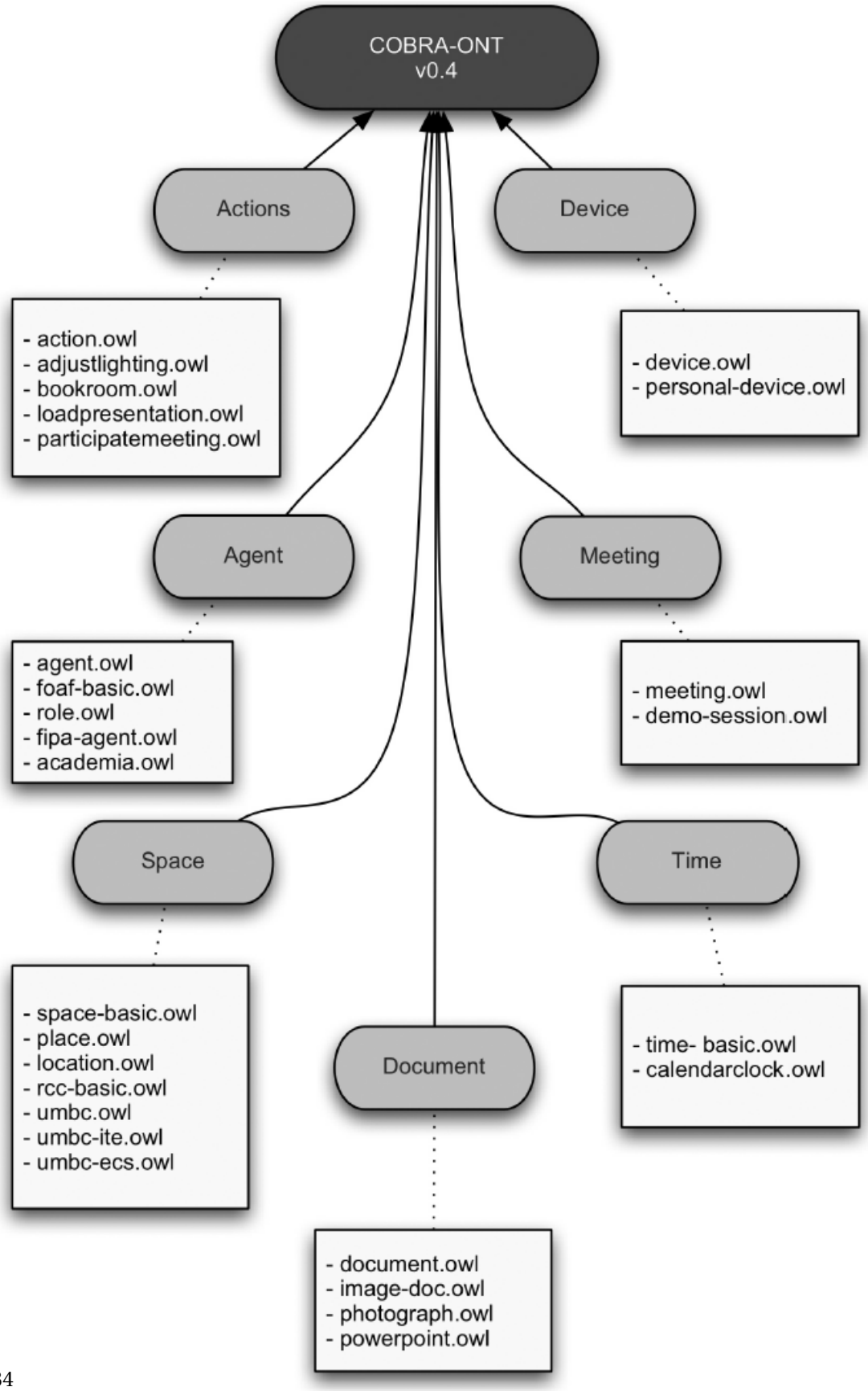


FIGURE 2.4 – Structure de CORBA-ONT v0.4 [CFJ04a]

2.3.2 Ontologie CONON

CONON (*Context Ontology*) [GWPZ04] est une ontologie de haut niveau pour modéliser le contexte dans un environnement pervasif. Elle fournit un modèle générique pour la représentation et la classification des informations de contexte. Elle [WZGP04] est décrite en langage OWL (*Ontology Web Language*) sous une forme hiérarchique à deux niveaux comme le montre la figure 2.5 :

— Haut niveau (**upper ontology**) : est une ontologie qui décrit l'ensemble de classes de base et qui sont communes à tous les domaines, telles que les informations de localisation (*Location*), les activités (*Activity*), les personnes (*Person*), et les entités de calcul (*CompEntity*). Chacune de ces classes est associée à des propriétés afin d'exprimer ses relations avec les autres classes. Par exemple, une activité se déroule (*locatedln*) dans une localisation et utilise une entité de calcul, ou une personne est propriétaire (*ownsCompEnt*) d'une entité de calcul, est engagée (*engagedln*) dans une activité, est située (*situatedln*) dans une localisation, et possède (*ownCompEnt*) une entité de calcul.

— Bas niveau (**Domain-specific ontology**) : représente une spécialisation de l'ontologie de haut niveau qui permet de décrire des contextes pertinents spécifiques à un domaine. Elle consiste en une collection de sous-ontologies qui étend les classes abstraites en classes plus spécifiques, permettant ainsi une forte extensibilité et capacité d'adaptation à différents domaines spécifiques.

En effet, les classes de base contiennent des sous classes qui présentent des extensions de plus haute spécificité et facilitent l'intégration des concepts propres à un domaine précis. Le langage OWL permet la structuration des entités de sous-classes de manière hiérarchique par la propriété *owl :subClasseOf*. Ainsi, la localisation est divisée en espace intérieur (*owl :Indoor*) et extérieur (*owl :Outdoor*). La classe activité (*owl :Activity*) distingue entre les activités programmées (*owl :SchedualedActivity*) et les activités déduites (*owl :DeducedActivity*). L'entité de calcul (*owl :CompEntity*) peut être un service (*owl :Service*), une application (*owl :Application*), un périphérique (*owl :Device*), un réseau (*owl :Network*), ou un agent (*owl :Agent*).
Enfin, CONON repose sur le framework Jena 2 [Rey04] pour la création de règles de raisonnement en logique du premier ordre. Elle intègre deux mécanismes de raisonnement sur le contexte : la détection et la correction des informations de contexte inconsistantes, et la déduction de contexte implicite de haut niveau à partir d'un contexte explicite de bas niveau (à l'aide de règles de symétrie, de transitivité et des règles définies par l'utilisateur).

2.3.3 Langage CoOL

CoOl (*Context Ontology Language*) [SLPF03] est un langage d'ontologie basé sur le modèle ASC (Aspect-Scale-Context). Ce modèle est nommé d'après ses concepts, qui sont l'aspect, l'échelle et les informations de contexte. Comme l'illustre la figure 2.6, chaque aspect est

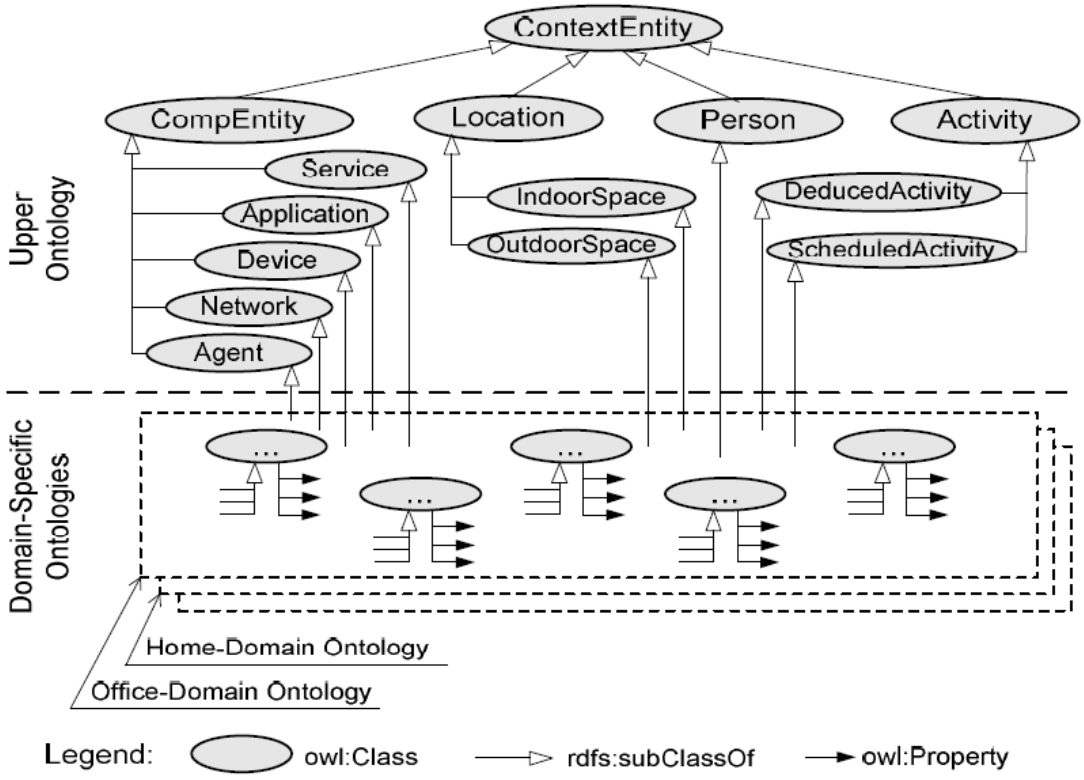


FIGURE 2.5 – Ontologie de contexte CONON [WZGP04]

composé d'un ou plusieurs échelles, et chaque échelle est également composé d'une ou plusieurs informations de contexte. Ces concepts fondamentaux sont liés par les relations *hasAspect*, *hasScale* et *constructedBy*. Par exemple, l'aspect *DistanceSpatiale* peut avoir deux échelles, l'échelle *Kilomètre* ou l'échelle *Mile* pour exprimer l'information de contexte *20*.

Le langage CoOl est une collection de plusieurs fragments, regroupés en deux parties :

— **Noyau CoOl** (*CoOl Core*) : représente une projection du modèle ASC, ce qui permet de classifier les informations de contexte selon l'échelle des données qu'elles représentent. Le noyau coOl est écrit en trois langages différents pour faciliter son utilisation par plusieurs types de plateformes : OWL [MVH+04] et DAML+OIL [HM00] qui sont deux langages des ontologies du web sémantique basés sur XML et RDF/S, et le langage F-Logic [KL89] qui combine les caractéristiques du paradigme orienté objet et de la logique de prédicats.

— **Schéma d'intégration CoOl** (*coOl Integration*) : représente un ensemble de protocoles qui permettent à des systèmes ou des services web d'utiliser le noyau de l'ontologie.

2.4 Approches Orientées Algèbres de Processus

Les algèbres de processus sont des langages abstraits conçus pour la spécification et l'étude du comportement des systèmes concurrents où les notions de hiérarchie, de modularité, de communication, d'abstraction et de parallélisme sont rigoureusement définies dans un formalisme mathématique qui supporte des algorithmes et outils de preuve. Dans cette section, nous décrivons un ensemble d'approches basées sur les algèbres des processus ayant comme objectif la modélisation de l'information contextuelle.

2.4.1 Calcul des Systèmes Communicants CCS

Le calcul des systèmes communicants ou CCS (*Calculus of Communication Systems*) [Mil80] a été proposé comme un langage de description des processus communicants (des agents dans la terminologie des Algèbres de Processus) où le mot agent désigne une machine, une mémoire, une procédure. La communication est un moyen d'interaction entre les agents. Dans la sémantique du calcul CCS, un agent est vu comme une boîte noire qui peut avoir un nom qui l'identifie, et dispose d'une interface d'échange d'événements ou d'actions avec son environnement. L'interface décrit l'ensemble de ports de communication, également appelés canaux, que l'agent peut utiliser pour interagir avec d'autres agents qui résident dans son environnement. Si une action n'est pas observable par l'environnement, elle est dite interne : dans ce cas, le processus a évolué de manière interne sans communication vers son environnement.

Le calcul CCS fournit une syntaxe concrète avec laquelle on peut décrire les processus concurrents, ainsi que le placement d'un processus donné dans un environnement :

$$P ::= 0 \quad | \quad x.P \quad | \quad \overline{x}.P \quad | \quad P + P' \quad | \quad P \mid P'$$

où 0 est le néant (ou l'inaction) processus qui n'a pas de comportement, $x.P$ accepte une entrée sur un canal appelé x, et procède ensuite un P, $\overline{x}.P$ fournit une sortie sur un canal appelé x, $P + P'$ est l'opérateur de choix qui se comporte soit comme P ou comme P', et $P|P'$ est la composition parallèle de deux processus exécutés simultanément. La syntaxe CCS offre un certain nombre d'autres constructions, mais elles sont omis pour plus de simplicité.

2.4.2 π-calcul

Le π-calcul (π-*calculus*) [MPW92] est un formalisme dérivé de CCS [Mil80] adapté à l'expression de la *mobilité* et de la *dynamicité* grâce à la notion de *nommage*. Il est considéré comme le fondement de la programmation concurrente. En particulier, il constitue une amé-

lioration significative en terme d'expressivité, en modélisant des agents qui se fonctionnent parallèlement en échangeant entre eux des messages via des canaux. Ainsi, il permet de décrire des systèmes de processus mobiles dont la topologie d'interaction varie au cours de l'exécution, c'est-à-dire, un agent peut créer un canal de communication sur lequel il peut envoyer ou recevoir des messages, puis le communiquer à ses pairs, qui acquièrent alors de nouvelles capacités de communication.

Formellement, un agent π-calcul peut prendre la forme suivante :

$$P ::= 0 \quad | \quad \alpha.P \quad | \quad P+Q \quad | \quad P \parallel Q \quad | \quad [x=y]P \quad | \quad [x \neq y] \quad | \quad !P \quad | \quad (\nu x)P$$

où les noms de canaux, notés $a, b, ..., z$, servent à indiquer à la fois le canal utilisé et la valeur transmise lors d'une communication entre deux agents.

La syntaxe du π-calcul est résumée dans le tableau 2.1.

Tableau 2.1 – Syntaxe du π-calcul

Préfixes	α	::=	$\overrightarrow{a}.\langle x \rangle$	Émission de la valeur, stockée dans la variable x, via le canal a
			$a.(x)$	Réception de la valeur, stockée dans la variable x, via le canal a
			η	Action interne, non observable
Processus	P	::=	0	Nul
			$\alpha.P$	Préfixion d'un processus P par une action α
			$P+Q$	Choix entre le processus P et le processus Q
			$P \parallel Q$	Composition parallèle de deux processus P et Q
			$[x=y]P$	Égalité
			$[x \neq y$	Inégalité
			$(\nu x)P$	Restriction
			$!P$	Réplication ou composition infinie de $P \parallel P \parallel ... \parallel P$
			$A(y_1, ..., y_n)$	Identifiant
Définitions			$A(x_1, ..., x_n) \stackrel{\text{def}}{=} P$	où $i \neq j \Rightarrow x_i \neq x_j$

2.4.3 Ambiants Mobiles

Le calcul des ambiants mobiles (*Mobile Ambients*) [CG98] décrit la migration de processus résidant dans des environnements dynamiques. Dans ce formalisme, un ambiant présente les caractéristiques principales suivantes :

— Un ambiant correspond à une boîte noire où la principale propriété est l'existence d'une frontière autour de lui.

— Chaque ambiant a un nom où le nom est utilisé pour contrôler l'accès (entrée, sortie, communication, etc.)

— Chaque ambiant possède une collection d'agents locaux (également appelés threads, processus, etc.)

— Chaque ambiant a une collection de sous-ambiants où chaque sous-ambiant a son propre nom, agents, sous-ambiants, etc.

— Les ambiants peuvent être imbriqués les uns dans les autres.

— Un ambiant peut se déplacer dans son ensemble.

— Les interactions n'impliquent que des processus dans des ambiants voisins.

— Les entités qui peuvent être communiquées sont soit des noms ou des capacités.

Á partir du tableau 2.2, nous pouvons noter qu'un processus ambiant p peut être le processus interne 0, une composition parallèle $P \mid Q$, un processus répliqué $!P$, un ambiant $n[P]$ nommé n, un processus $(\nu x)P$ avec un nom local unique x, ou un processus $C.P$ gardé par la capacité C. Le mécanisme le plus simple de communication que nous pouvons imaginer est la *communication locale anonyme* dans un ambiant (*ambiant I/O*), c'est-à-dire, la réception de message $(x).P$, ou l'envoi d'un message asynchrone $\langle C \rangle$. Les capacités représentent les actions de migration (in / out), la dissolution d'un ambiant (open), ou la création des chemins en combinant plusieurs capacités ($C.C'$). L'incapacité est présentée par ϵ.

Les différences principales du calcul des ambiants mobiles par rapport aux travaux antérieurs sur les calculs de processus sont les suivantes :

— L'existence des emplacements distincts est représentée par une topologie des frontières. Cette topologie induit une notion abstraite de la distance entre les emplacements. Les emplacements ne sont pas uniformément accessibles, et ne sont pas identifiés par des noms uniques.

— La mobilité des processus est représentée comme un franchissement des frontières. En particulier, la mobilité des processus n'est pas représentée comme la communication des processus.

— La sécurité est représentée comme la capacité ou l'incapacité à franchir les frontières. En particulier, la sécurité n'est pas représentée par des primitives cryptographiques ou des listes de contrôle d'accès.

Tableau 2.2 – Syntaxe des ambiants mobiles

Noms			$a, b, c, ..., n, m, p, q \in N$	Ensemble infini des noms
Variables			$w, x, y, z \in N$	Ensemble infini des variables
Processus	P	::=	0	Processus interne
			$P \mid Q$	Composition parallèle de deux processus P et Q
			$!P$	Réplication ou composition infinie de $P \mid P \mid ... \mid P$
			$n[P]$	Ambiant nommé n
			$(\nu x)P$	Restriction
			$C.P$	Processus gardé par la capacité C
			$(x).P$	Réception de message
			$\langle C \rangle$	Message asynchrone
Capacités	C	::=	in_n	Migration entrante
			out_n	Migration sortante
			open_n	Dissolution d'ambiant
			ϵ	Nulle
			$C.C'$	Chemin

— L'interaction n'implique que des processus dans des ambiants voisins, c'est-à-dire, les ambiants ayant des frontières communes.

2.4.4 Calculs d'Action

Les calculs d'action (*Action Calculi*) représentent une autre tentative de généraliser et d'unifier encore les modèles d'interaction. Beaucoup des idées présentées dans les bigraphes ont été initialement développées dans le cadre de calculs d'action, y compris la construction des termes à partir d'atomes, la syntaxe visuelle (graphe d'action) [Mil96], la sémantique de réécriture [Sew02], et le travail original qui a fait exposer les congruences de la bi-simulation [LM00b]. Ce formalisme peut être considéré comme un précurseur direct au développement des bigraphes dans la mesure où les bigraphes représentent l'extension et l'évolution des calculs d'action pour englober des idées plus générales, et la poursuite du développement pour permettre des solutions aux défis posés par l'informatique ubiquitaire.

2.5 Approches Orientées Systèmes Réactifs Bigraphiques

Les systèmes réactifs bigraphiques [Mil01b, Mil04, Mil09] sont un cadre formel permettant de décrire, à la fois, la distribution spatiale des agents et leurs interconnexions mobiles. La principale motivation qui a conduit à l'élaboration des systèmes réactifs bigraphiques est la modélisation et le raisonnement sur le comportement des systèmes ubiquitaires. Récemment, les systèmes réactifs bigraphiques sont devenus de plus en plus importants à cause de leur efficacité en termes de fournir un formalisme de modélisation graphique unifié qui intègre les différents avantages des algèbres de processus. Dans cette section, nous décrivons les approches les plus connues appartenant à cette catégorie, et le principe sur lequel se base chaque approche pour décrire les différents aspects des systèmes sensibles au contexte.

2.5.1 Modèle Platographique

Birkedal et al. [BDE+06] ont fait valoir que les systèmes réactifs bigraphiques sous leur forme naturelle (c'est-à-dire tels qu'ils ont été introduits par Robin Milner [Mil04]) ne sont pas adaptés à la modélisation des systèmes sensibles au contexte. Par conséquent, ils ont proposé des modèles dits platographiques (*platographical models*) qui sont des nouveaux modèles bigraphiques sophistiqués, où l'état et la dynamique des systèmes sensibles au contexte sont logiquement divisés en trois parties : le contexte actuel, le contexte observé (ou proxy), et les agents de calcul respectivement. Les agents de calcul et le contexte actuel sont séparés et interagissent uniquement par le proxy.
La définition formelle d'un modèle platographique est donnée comme suit :

Définition 2.1 (Modèle Platographique)**.** Un modèle platographique est un triple C, P, A des systèmes réactifs bigraphiques, de telle sorte que $M = C \cup P \cup A$ est lui-même un système réactif bigraphique et $C \perp A$. Un état du modèle est un bigraphe de M de la forme $/\overrightarrow{x}.(C \parallel P \parallel A)$ où $\mathbf{C} \in C$, $\mathbf{P} \in P$, $\mathbf{A} \in A$, et $/\overrightarrow{x}$ est un vecteur de noms.

— Les systèmes réactifs bigraphiques C, P et A indiquent respectivement : le contexte actuel, le contexte observé (ou le proxy), et les agents de calcul,

— $C \perp A$ indique que les systèmes réactifs bigraphiques C et A sont indépendants si et seulement si les signatures K_C et K_A sont disjointes,

— L'union $C \cup P \cup A = (K_C \cup K_P \cup K_A, R_C \cup R_P \cup R_A)$ quand $K_C \cup K_P \cup K_A$ s'entendent sur les arités des contrôles $K_C \cap K_P \cap K_A$.

Alors qu'un élément de **C** modélise un contexte, un élément de **P** modélise un modèle de ce contexte. La condition d'indépendance garantit que les agents ne peuvent que manipuler le proxy ; pas le contexte lui-même (la condition de l'indépendance assure la séparabilité).

Pour interroger ou modifier le contexte, les agents doivent utiliser le proxy en tant que capteur et actionneur.

2.5.2 Contexte et Capacités

L'approche proposée par Xu et al. [XXL11] est basée sur la technique de mapping entre les éléments des systèmes sensibles au contexte et de leurs représentations bigraphiques. Les éléments d'un système sensible au contexte sont regroupés en trois catégories : *contextes*, *calculs* et *capacité* où chaque catégorie dispose d'un contrôle et une représentation graphique unique.

Comme le montre la figure 2.7, le nœud elliptique *con* représente une entité de contexte. *cap* est un nœud hexagonale représentant une entité de capacité. Le nœud rectangulaire P représente une entité dans laquelle le calcul peut se produire. Enfin, les liens ouverts x et y représentent la connectivité entre les entités de contexte où les noms des liens sont utilisés pour distinguer les différents types de contextes.

Les principaux éléments d'une règle de réaction sont les entités de capacité dont chaque capacité cap peut changer un contexte cap qui la lie, trouver ses changements, puis disparaître.

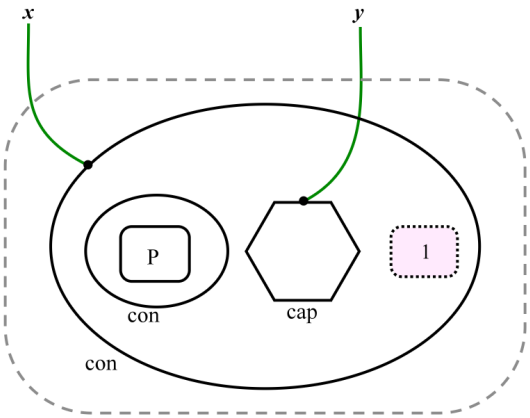


FIGURE 2.7 – contexte et capacités

2.5.3 Contexte et Actions

Dans le même ordre d'idées, Wang et al. [WXL11] ont introduit des contrôles et des représentations graphiques pour distinguer les différentes informations contextuelles : physiques, virtuelles et actions. Dans cette approche, trois catégories de nœuds ont été proposées : *elliptiques*, *rectangulaires* et *hexagonales*. Les nœuds elliptiques sont utilisés pour indiquer les entités physiques, telles que les personnes, les biens, les équipements, etc. Les nœuds

rectangulaires sont utilisés pour représenter les entités virtuelles, telles que les applications, les modes, statuts, etc. Enfin, Les nœuds hexagonales sont introduits pour représenter les actions. Le comportement de l'auto-adaptation des systèmes sensibles au contexte est modélisé comme une règle de réaction. Chaque règle de réaction est déclenchée par une *action* où cette dernière disparaît après que la règle de réaction est effectuée.

Le modèle de la structure des systèmes sensibles au contexte est représenté dans la figure 2.8.

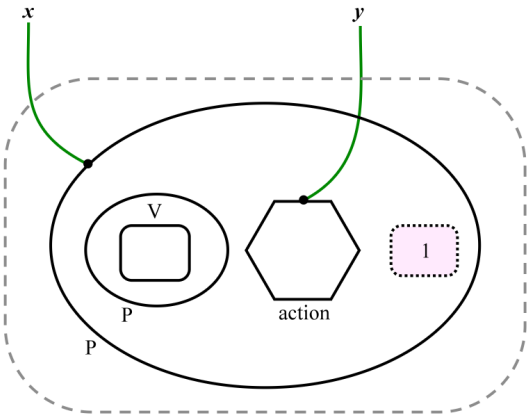


FIGURE 2.8 – Contexte et actions

2.5.4 BiAgent

Pereira et al. [PKS12] ont introduit un nouveau modèle de modélisation de systèmes sensibles au contexte appelé BiAgent (*Bigrapical Agents*) ou agents bigraphiques, qui fournit une séparation claire entre le monde physique et le monde virtuel, où le monde physique est modélisé par des bigraphes alors que les agents sont utilisés pour modéliser le monde virtuel. Dans ce modèle, les entités telles que les véhicules, les personnes, les ordinateurs ou les smart-phones sont des nœuds liés par le graphe de places ou le graphe de liens. Les agents modélisent les entités virtuelles telles que les calculs. Les agents peuvent observer le bigraphe, être héberger ou migrer dedans, et également le contrôler.

Le contrôle est un ensemble de règles de réaction telles qu'elles ont étés présentées par Robin Milner [Mil09]. Le modèle BiAgent ajoute des définitions pour l'observation, l'hébergement et la migration. L'observation est comme l'interrogation proposée par Birkedal [BDE$^+$06], mais définie de manière plus abstraite. La migration et l'hébergement sont plus particulièrement au modèle BiAgent. Un agent est simplement un calcul séquentiel, c'est-à-dire, le temps est linéaire. Á chaque fois que le temps avance, l'agent demande une observation, une migration, ou une action contrôle. Avec une équivalence de Nerode appropriée [Hop07],

l'agent dans le modèle BiAgent serait une machine d'état fini ou infini. L'agent mémoire peut être fini ou dénombrable.

La combinaison des agents et des bigraphes modélisant des systèmes de calcul embarqué dans un monde physique évolue comme suit :

— Les agents planifient des actions de contrôle dans le temps de l'agent qui sont des règles de réaction bigraphiques.

— Lorsque ces règles prennent effet dans le temps du bigraphe, le monde physique s'écoule d'un bigraphe à un autre entraînant des changements structurels.

— Lorsque le flux en temps du système réactif bigraphique reflète dans le temps de l'agent, la transition du bigraphe peut être observé par les agents et de nouvelles actions de migration ou de contrôle peuvent être planifiées à répéter le processus.

La définition formelle d'un BiAgent est comme suit [PKS12] :

Définition 2.2 (BiAgent)**.** Soit $a \bullet \mathscr{B}$ un BiAgent tel que :

$$a = (\mathscr{O}, \mathscr{U}, host_0, obs, ctr, mgrt)$$

et

$$B = (\mathbb{B}, \mathscr{R}, \mathbb{U}, B_0, F)$$

est un bigraphe sans mémoire fonctionnant sur une structure physique où :

— $\mathscr{O} \subseteq \mathbb{B}$ est l'espace d'observation.

— $\mathscr{U} = \mathscr{R}_a \times V_{\mathbb{B}}$ est l'espace de contrôle où $\mathscr{R}_a$ dénote un ensemble de règles de réaction.

— $host_0 \in V_{\mathbb{B}}$ est le nœud de bigraphe hébergeant l'agent.

— $obs : \mathbb{B} \times V_{\mathbb{B}} \to \mathscr{O}$ est une fonction d'observation.

— $ctr : \mathscr{O} \to \mathscr{U}$ est une fonction de contrôle donnant une observation qui produit une action de contrôle.

— $mgr : V_{\mathbb{B}} \times \mathscr{O} \to V_{\mathbb{B}}$ est une fonction de migration qui fournit l'hôte suivant.

Soit $(B_0, B_1, B_2, ...)$ la trace bigraphique de $\mathscr{B}$. Alors a peut effectuer l'une des opérations suivantes sur $\mathscr{B}$ à un temps donné i :

— Observer : $obs(B_i, h_i^a)$ où h_i^a dénote l'hôte de a au temps i.

— Migrer : $mgrt(h_i^a, o)$ fournit l'hôte suivant de a en considérant l'hôte actuel et une observation o.

— Agir : $u_i = ctr(o)$ fournit l'action de contrôle suivante pour une observation o. Ensuite, u_i est appliquée à la fonction de transition de $\mathscr{B}$ produisant $B_{i+1} = F(B_i, u_i)$.

Le modèle BiAgent a un modèle hybride de temps comme le modèle de système hybride Sifakis-Henzinger [Hen00] ou le modèle des automates temporisé Alur-Dill [AD94]. Le temps de bigraphe est appelé *temps physique*, car le changement des bigraphes modélise un changement dans un monde physique. L'exécution d'une migration ne modifie pas le bigraphe, mais seulement la relation entre un agent et le bigraphe, c'est-à-dire, la relation d'hébergement. L'avancement dans la relation d'hébergement entre les agents et le bigraphe est appelé un avancement dans le *temps du réseau*. Enfin, chaque agent est un calcul de temps linéaire qui avance par la planification de l'observation, le contrôle, ou les actions de migration, où le temps de l'agent est appelé le *temps de calcul*.

2.6 Synthèse et Évaluation

Cette section présente une synthèse des points spécifiques des différentes approches que nous venons d'étudier. Notre objectif était de fournir un aperçu des approches permettant de modéliser les informations contextuelles, et de définir les aspects des systèmes sensibles au contexte couverts par chacune de ces approches.
Les principaux critères d'évaluation sont les suivants :

— Niveau d'abstraction

— Type de communication

— Distribution spatiale

— Mobilité

— Mécanisme du raisonnement

— Partage de l'information contextuelle

— Réutilisation

Les approches orientées modèles reposent sur la méta-modélisation pour décrire des modèles de contexte qui peuvent être réutilisés par plusieurs applications. Les modèles décrits sont extensibles pour permettre aux concepteurs de définir de nouveaux types de relations entre les informations de contexte, et ainsi la possibilité d'intégrer des notions spécifiques à leurs besoins. Cependant, les approches orientées modèles souffrent du manque de sémantique formelle, ce qui rend difficile de s'assurer de la cohérence des informations de contexte, et de décrire le comportement des systèmes sensibles au contexte.

Les approches orientées ontologies sont des approches formelles qui consistent à utiliser les concepts de l'ontologie pour, non seulement la description sémantique du contexte, mais aussi pour le partage et la distribution des informations de celui-ci. Elles proposent une séparation entre les informations de contexte de base et les informations spécifiques à un domaine précis, ce qui permet de réutiliser et d'enrichir plus facilement l'ontologie. De plus, les

approches orientées ontologies utilisent des moteurs d'inférence pour déduire des nouveaux contextes à partir des informations collectées. Cependant, la conception d'une ontologie ou la réconciliation des ontologies hétérogènes développées de manière autonome pour un domaine d'application donné est une tâche difficile qui pose des problèmes d'interopérabilité.

Les approches basées sur les algèbres de processus sont des approches logiques caractérisées par un degré de formalité très élevé. Ces approches utilisent des sémantiques formelles pour modéliser le contexte dans le but de raisonner sur les informations collectées. Cependant, elles ne permettent pas de décrire la validité temporelle des informations de contexte ni les relations qui peuvent exister entre elles. Les approches appartenant à cette catégorie permettent de poser des bases de raisonnement rigoureuses sur les informations de contexte, mais leurs applications concrètes restent limitées ; notamment à cause de la complexité de description qu'engendre ce type d'approches.

Les approches basées sur les systèmes réactifs bigraphiques sont des approches prometteuses car elles se basent sur un formalisme qui offre une théorie unificatrice pour modéliser les différents aspects des systèmes sensibles au contexte. L'idée derrière les modèles platographiques [BDE+06] et BiAgents [PKS12] est de diviser logiquement l'état d'un système sensible au contexte et sa dynamique. Les auteurs ont fait valoir que cette séparation facilite non seulement le codage des systèmes sensibles au contexte en utilisant des langages spécifiques à des domaines, mais aussi aider efficacement à simuler ces systèmes. Par contre, la division d'un système sensible au contexte en différents systèmes réactifs bigraphiques ne conduit pas toujours à améliorer l'efficacité de la modélisation mais parfois elle conduit à une redondance inutile, surtout lorsqu'il s'agit de systèmes multi-états. C'est-à-dire, à chaque fois que le système se modifie, son modèle croît en taille, ce qui rend la modélisation prédictive plus en plus complexe. Les auteurs de [XXL11] et [WXL11] ont proposé deux approches basées sur la même idée, qui consiste à modéliser les systèmes sensibles au contexte en tant qu'ensemble d'entités hétérogènes (c'est-à-dire physique/contexte, virtuelle/calcul, action/-capacité). Chaque entité est un nœud explicitement identifié par un contrôle et une figure spécifiques. En outre, les auteurs ont introduit l'interaction entre une entité et une action/capacité pour modéliser structurellement le comportement des systèmes sensibles au contexte. Par conséquent, les approches proposées sont graphiquement riches d'analyser et de distinguer les différentes entités du système. Cependant, à chaque fois qu'une règle de réaction se produit, des nouveaux nœuds action/capacité sont créés et disparaissent simultanément fournissant des informations supplémentaires inutiles. En outre, les auteurs ont mis l'accent seulement sur la structure statique (graphique) de systèmes sensibles au contexte sans jeter une lumière sur la modélisation formelle de la dynamique du contexte.

Le tableau 2.3 récapitule les critères remplis par les différentes approches que nous avons étudiées.

Tableau 2.3 – Caractéristiques des approches étudiées

Famille		Composante de base	Abstraction	Communication	Distribution	Mobilité	Raisonnement	Partage	Réutilisation
Modèles	ContextUML	Classe UML	*	C/S	-	-	RD	-	+
	CML	Fait	+	C/S	-	-	LT	+	+
	MLContext	Entité	*	C/S	-	-	MI	-	+
Ontologies	COBRA-ONT	Agent	*	C/S	-	-	MI	+	+
	CONON	Classe OWL	*	PAP	-	-	MI	+	+
	CoOL	Aspect	*	PAP	-	-	MI	+	+
Algèbres de processus	CCS	Processus	+	H	-	-	RI	-	-
	π-calcul	Processus	+	H	-	-	RI	-	-
	Calcul d'Action	Processus	+	H	-	-	RI	-	-
	Ambiants Mobiles	Processus	+	H	+	-	RI	-	-
Bigraphes	Platographes	Bigraphe	*	C/S	+	+	RG	+	+
	Contexte et Capacités	Nœud	+	PAP	+	+	RG	+	+
	Contexte et Actions	Nœud	+	PAP	+	+	RG	+	+
	BiAgent	Agent	*	PAP	+	+	RG	+	+

+ : Disponible
- : Non disponible
* : Haut Niveau
C/S : Client/Serveur
H : Handshaking
PAP : Pair-à-Pair
LT : Logique Ternaire
RI : Règles d'inférence
MI : Moteur d'inférence
RD : Raisonnement Diagrammatique
GR : Raisonnement Graphique

2.7 Conclusion

Le but de ce chapitre était de dresser un état de l'art des travaux effectués sur la modélisation des différents aspects structurels et comportementaux dans le cadre des systèmes sensibles au contexte. Nous avons d'abord présenté plusieurs approches de modélisation de ces systèmes : les approches orientées modèles, les approches orientées ontologies, les approches orientées algèbres de processus, et les approches orientées systèmes réactifs bigraphiques. L'objectif de l'étude que nous avons menée consiste à montrer l'apport de chaque approche de modélisation, leur pertinence et leurs limites. Après avoir analysé ces approches, nous avons conclu que les systèmes réactifs bigraphiques sont les mieux adaptés pour décrire des systèmes répondant aux différents critères d'ubiquité à savoir la sensibilité au contexte, la mobilité, la distribution, la dynamicité.

Dans le chapitre suivant, nous présentons un état de l'art sur les systèmes réactifs bigraphiques. Nous commençons par introduire l'anatomie de ces systèmes, leurs concepts généraux et la terminologie propre à cet formalise. Ensuite, nous présentons la dynamique des systèmes réactifs bigraphiques ainsi que leur algèbre de termes. Enfin, nous étudions les différents outils développés autour de ce formalisme.

FIGURE 2.6 – Modèle ASC [SLPF03]

3 Systèmes Réactifs Bigraphiques

Sommaire

3.1 Introduction

La théorie de la *concurrence* a langui pendant un certain temps, avec un regain d'intérêt au début des années 1970 avec le développement du modèle d'acteur (*Actor Model*) [Agh86] dans lequel les composants principaux d'un système sont des acteurs qui envoient et reçoivent des messages. En effet, la *concurrence* est l'hypothèse clé qui sous-tend le développement des bigraphes et beaucoup d'autres formalismes tels que le Calcul des Systèmes Communicants (*Calculus of Communicating Systems* CCS) [Mil80], les Processus Communicants Séquentiels (*Communicating Sequential Processes* CSP) [H$^+$85], l'Algèbre de Processus Communicants (*Algebra of Communicating Processes* ACP) [BK86], π-calcul ((pi-calculus) [MPW92], la machine chimique abstraite de Berry et Boudol (*Chemical Abstract Machine* cham) [BB89], les Calculs d'Action (*Action Calculi*) [Mil93] et les Ambiants Mobiles (*Mobile Ambients*) [CG98].

La théorie de bigraphes a été développé par Robin Milner [Mil09]. Les bigraphes et leurs systèmes réactifs ont été développés comme un modèle graphique de calcul qui met l'accent, à la fois, sur la localité et la connectivité mobiles. La théorie a été développé avec deux objectifs principaux : (1) pour être en mesure d'intégrer dans le même formalisme les aspects importants des systèmes ubiquitaires ; (2) de fournir une unification des théories existantes en développant une théorie générale, dans laquelle les différents calculs existants pour la concurrence et la mobilité, en particulier le calcul des systèmes communicants [Mil80], le π-calcul [MPW92], le calcul ambiant [Mil93] et les réseaux de Pétri, peuvent être représentés, avec une théorie comportementale uniforme. Cette dernière est obtenue en représentant la dynamique des bigraphes par une définition abstraite de règles de réaction à partir de laquelle un système de transition peut être dérivé d'une manière telle qu'une relation de bi-simulation associée soit une relation de congruence.

L'objectif principal de ce chapitre est de fournir aux lecteurs qui ne sont pas familiers de bigraphes les informations indispensables à une bonne compréhension du contenu de la thèse. Nous commençons, tout d'abord, par introduire l'anatomie des bigraphes, leurs concepts généraux, la notation et la terminologie propres à cet formalise dans la section 3.2. Ensuite, la section 3.3 est consacrée à l'introduction des opérations fondamentales, que nous pouvons les effectuer sur les bigraphes, avec des exemples illustratifs. La partie de la théorie portant sur l'évolution dynamique de bigraphes est présentée dans la section 3.4. La section 3.5 introduit l'algèbre de termes des systèmes réactifs bigraphiques. Nous présentons, enfin, un ensemble d'outils développés autour de ce formalisme à la section 3.6.

Les définitions de ce chapitre sont tirées du livre de Robin Milner [Mil09], ses articles [Mil05, Mil06, Mil08] et rapports techniques [Mil01a, JM04] publiés. La plupart de ce contenu est simplement une distillation des excellentes ressources que nous avons pu puiser dans le cadre de l'apprentissage des bigraphes.

3.2 Anatomie des Bigraphes

Un bigraphe est la combinaison de deux structures indépendantes (voir figure 3.1) : le graphe de places et le graphe de liens [Mil09]. L'intersection entre ces deux graphes est un ensemble commun de nœuds, correspondant aux entités physiques ou virtuelles d'une application distribuée à code mobile.

Le graphe de places (figure 3.2) a la structure d'une forêt, présentant la distribution géographique de l'application. Le graphe de liens (figure 3.3) est un hypergraphe montrant le schéma de connectivité des différents nœuds. Alors qu'un arc dans le graphe de places montre la relation d'imbrication entre les éléments de l'application, un hyperarc dans le graphe de liens établit une connexion entre les ports de ces éléments. Chaque arbre dans le graphe de places représente une région qui peut contenir des sites, correspondant aux feuilles de l'arbre, où d'autres bigraphes peuvent être insérés ou hébergés.

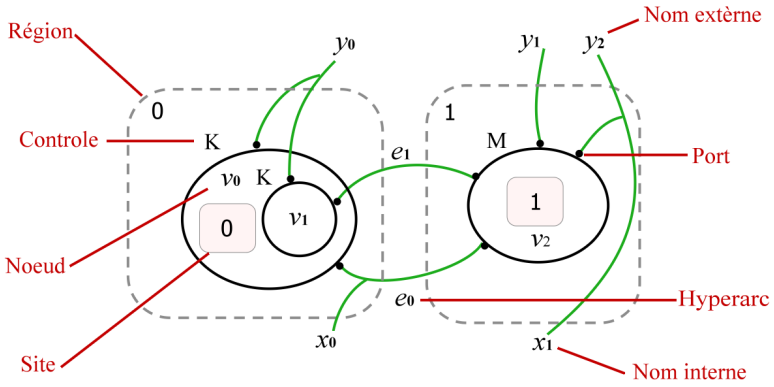


FIGURE 3.1 – Anatomie d'un bigraphe

Définition 3.1 (Bigraphe)**.** Formellement, un bigraphe [Mil09] est défini par :

$$G = (V, E, ctrl, G^P, G^L) : I \to J$$

— V est un ensemble fini de nœds qui peuvent être imbriqués.

— E est un ensemble fini d'hyperarcs.

— $ctrl : V \to \mathcal{K}$ est une transformation qui associe à chaque nœud $vi \in V$ un contrôle $k \in \mathcal{K}$ indiquant le nombre de ports et son comportement dynamique.

— $\mathcal{K}$ est une signature définie comme un ensemble de contrôles.

— $G^P = (V, prnt, ctrl) : m \to n$ est le graphe de places associé à G, où $prnt : m \uplus V \to V \uplus n$ est une fonction de parenté qui associe à chaque nœud ou site son parent hiérarchique.

— $G^L = (V, E, ctrl, link) : X \to Y$ est le graphe de liens de G, où $Link : P \uplus X \to E \uplus Y$ est une transformation.

— $I = \langle m, X \rangle$ et $J = \langle n, X \rangle$ représentent respectivement les interfaces internes et externes du bigraphe G, où m est le nombre de sites, c'est-à-dire, les emplacements dans le bigraphe susceptibles d'intégrer de nouveaux éléments, X est l'ensemble des noms internes, n est le nombre de régions, c'est-à-dire, les éléments du bigraphe indépendants les uns des autres (mais restent en liaisons) pouvant s'intégrer dans d'autres bigraphes et Y est l'ensemble des noms externes.

Pour illustrer les notations bigraphiques, la figure 3.1 représente un bigraphe

$$G = \langle 2, \{x_0, x_1\} \rangle \rightarrow \langle 2, \{y_0, y_1, y_2\} \rangle$$

où les ensembles de nœuds et d'arêtes sont donnés par $V = \{v_0, v_1, v_2\}$ et $E = \{e_0, e_1\}$ respectivement. $\mathcal{K} = \{K : 2, M : 4\}$ représente la signature du bigraphe G. $I = \langle 2, \{x_0, x_1\} \rangle$ est l'interface interne de G dans laquelle 2 est un ordinal représentant le nombre fini de sites et $X = \{x_0, x_1\}$ est l'ensemble des noms internes. De même, l'interface externe de G est donnée par $J = \langle 2, \{y_0, y_1, y_2\} \rangle$ où 2 représente le nombre de régions et $Y = \{y_0, y_1, y_2\}$ est l'ensemble des noms externes. Enfin, $G^P : 2 \rightarrow 2$ (figure 3.2) est le graphe de places de G, alors que $G^L : \{x_0, x_1\} \rightarrow \{y_0, y_1, y_2\}$ (figure 3.3) est le graphe de liens.

3.2.1 Graphe de Places

Le graphe de places [Mil06] se compose d'une forêt d'arbres, dont chacun a une racine. Il est utilisé pour représenter les notions de localité ou d'endiguement. Les arbres sont généralement établis comme dans la figure 3.2 avec la racine au sommet, et les feuilles (les nœuds) en bas.

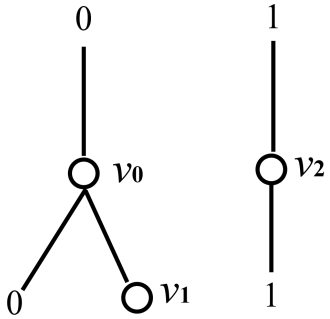


FIGURE 3.2 – Graphe de places

Définition 3.2 (Graphe de places)**.**

$$G^P = (V, prnt, ctrl) : m \rightarrow n$$

— V est un ensemble fini de nœuds.

— $prnt : m \uplus V \to V \uplus n$ est une fonction de parenté qui associe à chaque nœud ou site son parent hiérarchique. Nous utilisons la notation $m \uplus V$ pour signifier $\{0, ..., m\} \uplus V$ où les deux ensembles sont conservés disjoints.

— $ctrl : V \to \mathcal{K}$ est une transformation qui associe à chaque nœud $vi \in V$ un contrôle $k \in \mathcal{K}$ indiquant le nombre de ports et son comportement dynamique.

— La notation $m \to n$ permettant d'enregistrer les interfaces du graphe de places, où m et n représentent respectivement les interfaces internes et externes du graphe de places.

3.2.2 Graphe de Liens

Structurellement, le graphe de liens [Mil06] est un hypergraphe, ce qui signifie que chaque arc relie zéro ou plusieurs nœds où les liens sont non orientés, de telle sorte que la connexion de A à B connecte également B à A d'une manière symétrique. En outre, le graphe de liens capture la connectivité des éléments du bigraphe, ignorant leur imbrication. Cependant, il y a une analogie formelle proche entre les théories du graphe de places et graphe de liens (définition 3.3).

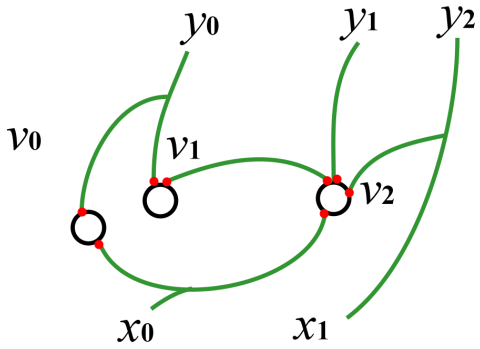


FIGURE 3.3 – Graphe de liens

Définition 3.3 (Graphe de liens)**.**

$$G^L = (V, E, ctrl, link) : X \to Y$$

— V est un ensemble fini de nœuds.

— E est un ensemble fini d'hyperarcs.

— $ctrl : V \to \mathcal{K}$ est une fonction de contrôle.

— $Link : P \uplus X \to E \uplus Y$ est une transformation montrant le flux de données des noms internes X ou les ports P vers les noms externes Y ou les hyperarcs E, où l'ensemble de ports P est défini par $P \stackrel{\text{def}}{=} \{(v, i) | v \in V \wedge i \in ar(ctrl(v))\}$, c'est-à-dire, un port est représenté par une paire de nœud $v \in V$ et un indice i.

3.2.3 Notation

Nous interprétons souvent un nombre naturel comme un ordinal fini, nommément $m = 0, 1, ..., m-1$. Nous écrivons $S \uplus T$ pour indiquer l'union des ensembles connus ou supposés être disjoints. Si la fonction f a un domaine S et $S' \subseteq S$, alors $f \restriction S'$ désigne la restriction de f vers S'. Pour deux fonctions f et g ayant des domaines disjoints S et T, nous écrivons $f \uplus g$ pour la fonction ayant le domaine $S \uplus T$ de telle sorte que $(f \uplus g) \restriction S = f$ et $(f \uplus g) \restriction T = g$. Nous écrivons id_S pour exprimer la fonction de l'identité de l'ensemble S. Dans la définition des bigraphes, nous supposons que les noms internes, les identifiants des nœuds et des hyperarcs sont tirés de trois ensembles finis, respectivement X, V et E, disjoints les uns des autres. On note les identifiants par des lettres minuscules : x, y, z pour les noms internes, v, u pour les nœuds, et $e_0, e_1, ...$ pour les hyperarcs. les lettres majuscules $A, B, ...$ sont utilisées pour désigner les bigraphes. Nous appelons l'interface triviale $\epsilon \stackrel{\text{def}}{=} \langle 0, \emptyset \rangle$ *l'origine*.

3.3 Opérations sur les Bigraphes

Dans cette section, nous montrons, à travers des exemples comment construire des bigraphes complexes à partir des bigraphes plus simples en appliquant les opérations de composition et de produit tensoriel [Mil05].

3.3.1 Composition

Algorithmiquement, la composition est le placement d'un bigraphe F dans un contexte représenté par un autre bigraphe G. La composition de deux bigraphes $G \circ F$ est définie si est seulement si l'interface externe de F apparie l'interface interne de G, et elle se fera via un hébergement des régions du bigraphe F dans les sites du bigraphe G, et une fusion des noms externes de F avec les liens de G qui ont des noms internes correspondants.

Définition 3.4 (composition de deux graphes de places)**.** Si $F : k \to m$ et $G : m \to n$ sont deux graphes de places avec des supports disjoints, leur composite

$$G \circ F = (V, ctrl, prnt) : k \to n \quad \text{(figure 3.4)}$$

a l'ensemble de nœuds $V = V_F \uplus V_G$ et la fonction contrôle $ctrl = ctrl_F \uplus ctrl_G$. Sa fonction de parenté $prnt$ est définie par : Si $w \in k \uplus V$ est un site ou un nœud dans $G \circ F$ alors

$$prnt(w) \stackrel{\text{def}}{=} \begin{cases} prnt_F(w) & \text{si } w \in k \uplus V_F \text{ et } prnt_F(w) \in V_F, \\ prnt_G(j) & \text{si } w \in k \uplus V_F \text{ et } prnt_F(w) = j \in m, \\ prnt_G(w) & \text{si } w \in V_G. \end{cases}$$

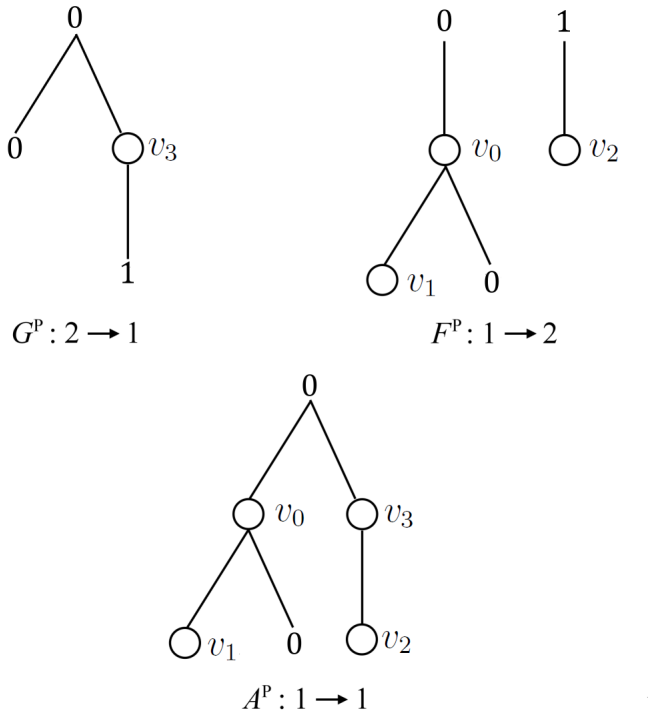


FIGURE 3.4 – Composition de deux graphes de places : $A^P = G^P \circ F^P$

Définition 3.5 (Composition de deux graphes de liens)**.** Si $F : X \to Y$ et $G : Y \to Z$ sont deux graphes de liens avec des supports disjoints, leur composite

$$G \circ F = (V, E, ctrl, link) : X \to Z \quad \text{(figure 3.5)}$$

a un ensemble de nœuds $V = V_F \uplus V_G$, un ensemble d'hyperarcs $E = E_F \uplus E_G$ et une fonction de contrôle $ctrl = ctrl_F \uplus ctrl_G$. Sa fonction $link$ est définie par : Si $q \in X \uplus P_F \uplus P_G$ est un point de $G \circ F$ alors

$$link(q) \stackrel{\text{def}}{=} \begin{cases} link_F(q) & \text{si } q \in X \uplus P_F \text{ et } link_F(q) \in E_F, \\ link_G(y) & \text{si } q \in X \uplus P_F \text{ et } link_F(w) = y \in Y, \\ link_G(q) & \text{si } q \in P_G. \end{cases}$$

Définition 3.6 (composition de deux bigraphes)**.** Si $F : \langle k, X \rangle \to \langle m, Y \rangle$ et $G : \langle m, Y \rangle \to \langle n, Z \rangle$ sont deux bigraphes avec des supports disjoints, leur composite est donné par :

$$G \circ F \stackrel{\text{def}}{=} (G^P \circ F^P, G^L \circ F^L) : \langle k, X \rangle \to \langle n, Z \rangle \quad \text{(figure 3.6)}$$

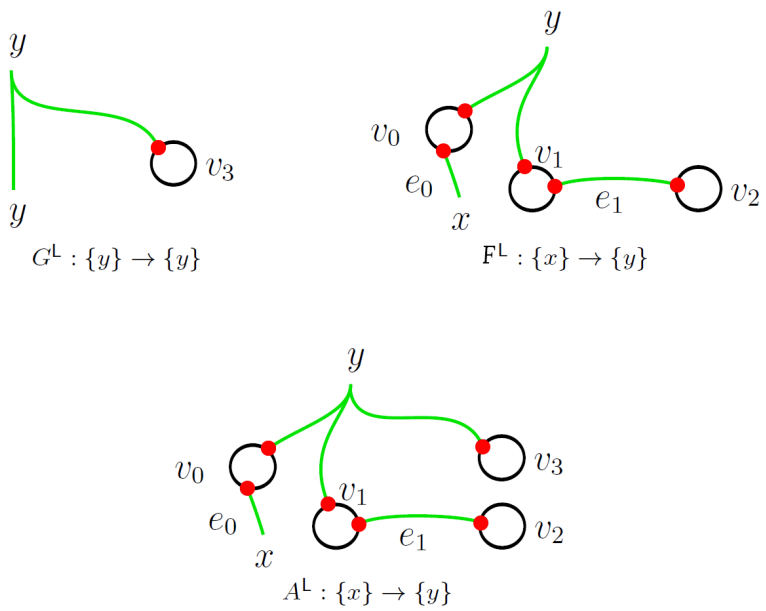


FIGURE 3.5 – Composition de deux graphes de liens : $A^L = G^L \circ F^L$

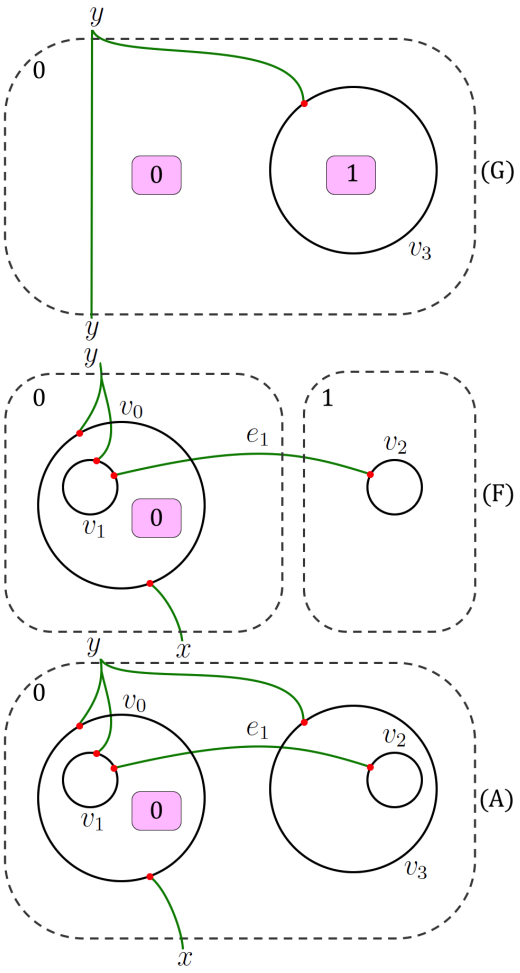


FIGURE 3.6 – Composition de bigraphes : $A = G \circ F$

3.3.2 Produit Tensoriel

Bien que l'insertion d'un bigraphe dans un autre pourrait être considérée comme la notion « normale » de la composition, il y a un autre moyen de composer les bigraphes qui est le produit de tensoriel (la composition horizontale). Le produit tensoriel est la juxtaposition de régions en joignant les liens ouverts communs.

Définition 3.7 (Produit tensoriel de deux graphes de places)**.** Si $G = (V_G, ctrl_G, prnt_G) : m_G \to n_G$ et $F = (V_F, ctrl_F, prnt_F) : m_F \to n_F$ sont deux graphes de places disjoints, leur produit tensoriel est donné par :

$$G \otimes F \stackrel{\text{def}}{=} (V_G \uplus V_F, ctrl_G \uplus ctrl_F, prnt_G \uplus prnt'_F) : (m_G + m_F) \to (n_G + n_F) \quad \text{(figure 3.7)}$$

où $prnt'_F(m_G + i) = n_G + j$ quand $prnt_F(i) = j$.

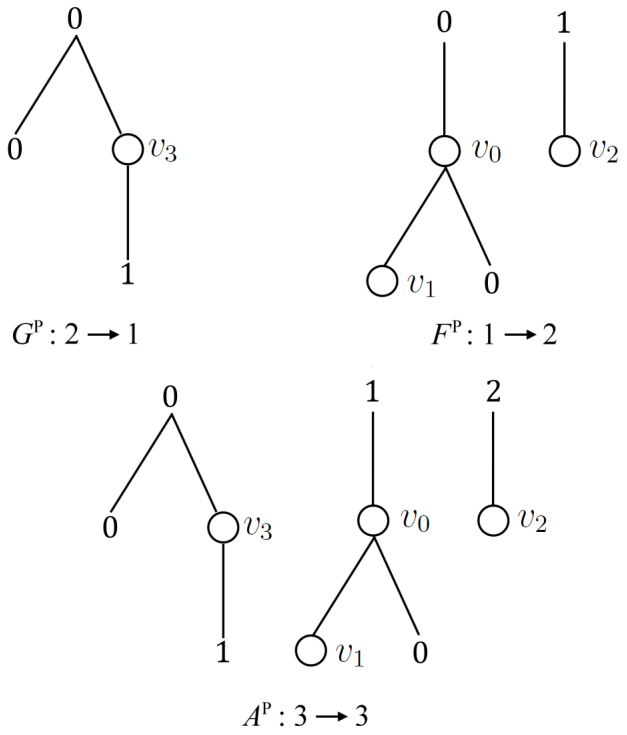


FIGURE 3.7 – Produit tensoriel de deux graphes de places $A^P = G^P \otimes F^P$

Définition 3.8 (Produit tensoriel de deux graphes de liens)**.** Si $G = (V_G, E_G, ctrl_G, link_G) : X_G \to Y_G$ et $F = (V_F, E_F, ctrl_F, link_F) : X_F \to Y_F$ sont deux graphes de liens disjoints, leur produit tensoriel est donné par :

$$G \otimes F \stackrel{\text{def}}{=} (V_G \uplus V_F, E_G \uplus E_F, ctrl_G \uplus ctrl_F, link_G \uplus link_F) : (X_G \uplus X_F) \to (Y_G \uplus Y_F) \quad \text{(figure 3.8)}$$

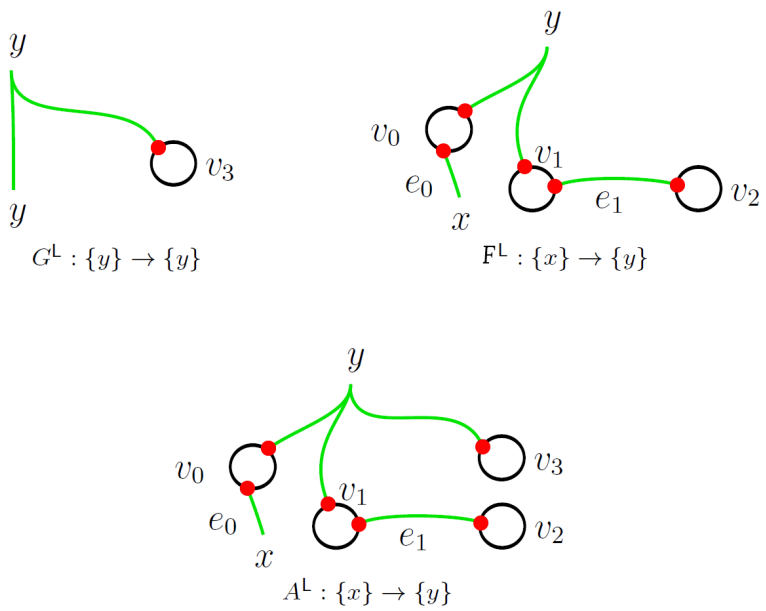


FIGURE 3.8 – Produit tensoriel de deux graphes de liens $A^L = G^L \otimes F^L$

Définition 3.9 (Produit tensoriel de deux bigraphes)**.** Si $F : \langle m_F, X_F \rangle \to \langle n_F, Y_F \rangle$ et $G : \langle m_G, X_G \rangle \to \langle n_G, Y_G \rangle$ sont deux bigraphes avec des supports disjoints, leur produit tensoriel est donné par :

$$G \otimes F \stackrel{\text{def}}{=} (G^P \otimes F^P, G^L \otimes F^L) : \langle m_G + m_F, X_G \biguplus X_F \rangle \to \langle n_G + n_F, Y_G \biguplus Y_F \rangle \quad \text{(figure 3.9)}$$

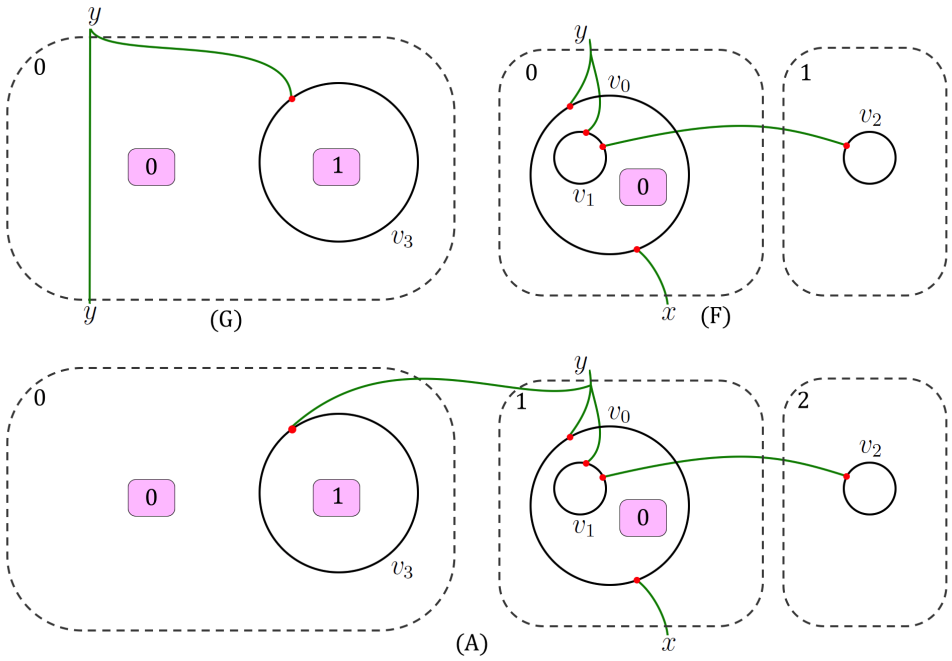


FIGURE 3.9 – Produit tensoriel de deux bigraphes : $A = G \otimes F$

3.4 Dynamique des Bigraphes

Les bigraphes sont équipés par une sémantique dynamique grâce aux *règles de réaction* qui sont semblables aux règles de réécriture de graphes [Mil06]. Cependant, les règles de réaction bigraphiques diffèrent des règles de réécriture habituelles en ce qu'elles sont paramétriques, c'est-à-dire une partie du contexte devient un paramètre qui peut être écarté ou dupliqué par la règle. En outre, les réactions sont générées de manière algébrique, contrairement aux approches *single-pushout* ou *double-pushout* de réécriture de graphes.

Chaque règle de réaction bigraphique consiste en un redex et un reactum. Le *redex* est une pré-condition pour la réaction, représentée par un pattern d'imbrication et/ou liaison, alors que le *reactum* est une post-condition indiquant comment la réaction changera ce pattern. Les endroits où les réactions peuvent se produire sont déterminés par les contrôles.

Dans ce contexte, on distingue deux types de transformations :

— Transformation sur les places représentant l'ajout, suppression, duplication ou déplacement d'une entité.

— Transformation sur les liens représentant la connexion ou la déconnexion d'un nœud à travers l'une de ses interfaces internes.

Définition 3.10 (Règle de réaction)**.** Une règle de réaction a la forme

$$\mathsf{R} = (R : m \to J, R' : m' \to J, \eta) \quad ou \quad R \to R'$$

où $R : m \to J$ est un bigraphe appelé redex (le pattern à changer), $R' : m' \to J$ est également un bigraphe appelé reactum (le nouveau pattern) et l'instanciation $\eta : m' \to m$ est une transformation d'ordinaux.

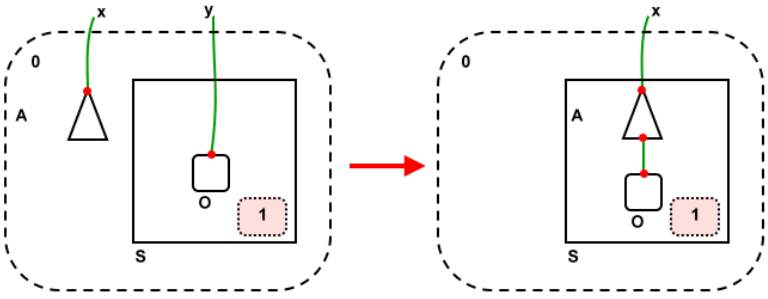


FIGURE 3.10 – Un exemple d'une règle de réaction

Par exemple, la règle de réaction présentée dans la figure 3.10 permet à un agent (A), dans la même région qu'une salle (S), d'entrer dans la salle et se connecter à l'ordinateur (O), c'est-à-dire, qu'avant l'exécution de la règle, l'agent (A) est en dehors de la salle et après l'exécution de la règle, il est dans la salle (S).

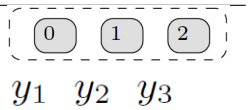
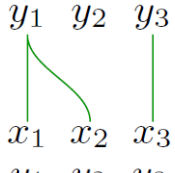
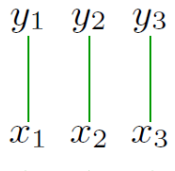
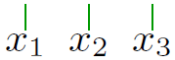
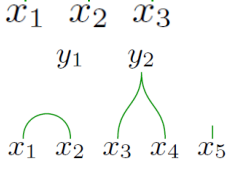
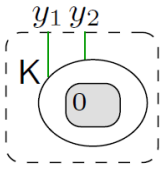
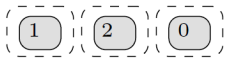
Formellement, cette règle de réaction est donnée par :

$$(R : 1 \to \langle 1, \{x, y\}\rangle, R' : 1 \to \langle 1, \{x\}\rangle, [1 \to 1])$$

3.5 Forme Algébrique

Dans cette section, nous introduisons un langage de termes [Mil08] pour représenter les systèmes réactifs bigraphiques qui doit être familier à la plupart aux lecteurs habitués à la notation des algèbres de processus. Il y a un certain nombre de bigraphes abstraits de base qui, avec la composition et le produit tensoriel, nous permettent de construire un bigraphe quelconque, c'est-à-dire, ils fournissent un langage de termes complet pour les bigraphes. Les bigraphes abstraits de base sont énumérés dans le tableau 3.1, où nous donnons également des exemples et nous définissons les noms et les variables que nous allons utiliser pour représenter chaque type de bigraphe abstrait de base.

Tableau 3.1 – Langage de termes bigraphiques

Notation		Exemple	
Fusion (*merge*)	$merge_n : n \to 1$	$merge_3 =$	
Substitution (σ)	$\overrightarrow{y} / \overrightarrow{X} : X \to Y$	$[y_1, y_2, y_3] / [\{x_1, x_2\}, \{\}, \{x_3\}] =$	
Renommage (α, β)	$\overrightarrow{y} / \overrightarrow{x} : X \to Y$	$[y_1, y_2, y_3] / [\{x_1, x_2, x_3\}] =$	
Clôture (*closure*)	$/X : X \to \{\}$	$/\{x_1, x_2, x_3\}$	
Câblage (ω)	$(id \otimes /Z)\sigma : X \to Y$	$id_{\{y_1, y2\}} \otimes \{z_1, z_2\} [y_1, z_1, y_2, z_2] /$ $[\{\}, \{x_1, x_2\}, \{x_3, x_4\}, t\{x_5\}] =$	
Ion	$K_{\overrightarrow{y}} : 1 \to \langle 1, \overrightarrow{y}\rangle$	$K_{[y_1, y_2]} =$	
Permutation (π)	$\{i \longmapsto j, ...\} : m \to m$	$\{0 \longmapsto 2, 1 \longmapsto 0, 2 \longmapsto 1\} =$	

Par exemple, l'expression algébrique correspondante à la règle de réaction présentée dans la figure 3.10 est comme suit :

$$x/A_x|S.(y/O_y|d_1) \rightarrow S.(x/A_x|O|d_1)$$

3.6 Outils Pratiques

Les systèmes réactifs bigraphiques sont un formalisme de haut niveau qui permet la modélisation des systèmes ubiquitaires. Récemment, des environnements de manipulation des systèmes réactifs bigraphiques ont vu le jour particulièrement, grâce au projet BPL (*Bigraphical Programming Language Project*) du laboratoire LaCoMoCo (*Laboratory for Context-dependent Mobile Communication*) [LaC04] tels que : BPL Tool [HG11, GDBH11], Big Red [FPH13], BigMC (Bigraphical Model Checker) [PDH12], et BigraphER [Sev12].

3.6.1 BPL Tool

BPL Tool [HG11, GDBH11] est une des premières implémentations qui avait fait des progrès significatifs pour faire avancer la théorie des systèmes réactifs bigraphiques. Cet outil consiste en un parseur, un moteur de matching, une interface Web, et une interface en ligne de commande permettant la manipulation, la simulation et la visualisation des systèmes réactifs bigraphiques. Le langage utilisé dans l'outil BPL Tool est appelé BPLL, il se compose d'un ensemble de blocs de langage Standard ML (SML) [Mil97] qui permettent d'écrire le BPLL directement dans SML.

L'outil BPL Tool a été utilisé pour modéliser les systèmes suivants :

— Protocole GeoCast [Nis]

— Protocole ARAN [Ben07]

— IEEE 802.11 MAC 4-way handshake [Ben07]

— Problème d'initié [Ben07]

— WS-BPEL and HomeBPEL [BGH$^+$12]

— Modèles Platographiques [Els09]

— Système de téléphonie mobile [GDBH11]

L'outil est disponible sur le site http://www.itu.dk/research/pls/wiki/index.php/BPL_Tool avec le guide d'installation et la documentation de l'API.

Termes et Opérations

La définition des signatures dans l'outil BPL Tool est centrée autour des contrôles. Pour définir un contrôle appelé K qui a le statut $s \in \{\text{active}, \text{passive}, \text{atomic}\}$, une arité globale m et une arité locale n on écrit :

$$val \quad K = s("K" =: m \to n)$$

Les bigraphes sont construits à partir des bigraphes élémentaires et des opérateurs présentés dans le tableau 3.2.

Tableau 3.2 – Langage de termes de BPL Tool

Ions :	(K, L, M : control; x, y, p, p′ : string)
K[y,...]	Ion avec un contrôle d'une arité globale
L[y,...][[x,...],...]	Ion avec un contrôle d'une arité globale/locale
K[p == y,...]	Ion avec un contrôle d'une arité globale et des ports nommés
L[p == y,...][p′ == x,...]	Ion avec un contrôle d'une arité globale/locale et des ports nommés
Connexions :	(x, y : string)
x/y	Renommage de lien
y//[x,...]	Substitution de lien
y//[]	Introduction d'un nom
−/x	Clôture d'un arc
−//[x,...]	Clôture multiple
idw[x,...]	Identifiant de connexion
Opérateurs :	(x : string; A, B : bgval; P : primebgval)
< [x,...] > P	noms abstraits x,... d'un premier P
A ∗ B	Produit tensoriel
A ‖ B	Produit parallèle
A′|′B	Produit premier
∗ ∗ [A,...]	Produit tensoriel de n facteurs
|||[A,...]	Produit parallèle de n facteurs
′|′′[A,...]	Produit premier de n facteurs
A ∘ B	Composition

Pour distinguer les termes bigraphiques relatifs à certaines formes, l'outil BPL Tool utilise les types suivants :

— bgterm : un terme qui pourrait ne pas être bien formé, par exemple : les interfaces dans des compositions peuvent ne pas correspondre entre elles.

— bgval : un terme bien formé avec des interfaces.

— 'a bgbdnf :un terme qui est sous une forme de liaison discrète indiquée par le type fantôme utilisé pour 'a :

  — M : molécule
  — S : nœud singulier de plus haut niveau
  — G : premier global discret
  — N : nom discret premier
  — P : premier discret
  — D : bigraphe discret
  — B : bigraphe
  — DR : bigraphe discret, régulier
  — BR : bigraphe régulier

L'outil BPL Tool propose un ensemble d'opérations formelles sur les systèmes réactifs bigraphiques présentées dans le tableau 3.3

Tableau 3.3 – Les opérations de BPL Tool

Égalité	
===	bgval ∗ bgval → bool
====	'abgbdnf ∗' abgbdnf → bool
Normalisation	
norm_v	bgval → Bbgbdnf
Dénormalisation	
denorm_b	'abgbdnf → bgval
Régularisation	
regl_v	bgval → BRbgbdnf
regl_b	Bbgbdnf → BRbgbdnf
Simplification	
simpl_v	bgval → bgval
simpl_b	'abgbdnf → bgval
Matching	
match_v	agent : bgval, redex : bgval → matchlazylist
match_b	agent : Bbgbdnf, redex : Bbgbdnf → matchlazylist
Règles de réaction :	(R, R′ : rule; rho : instantiation; N : string)
R – –| > R′	Règle avec un redex R, un reactum R' et une instanciation par défaut
R – rho – | > R′	Règle avec un redex R, un reactum R' et une instanciation rho
N ::: R – –| > R′	Règle nommée

Exemple : Système de Téléphonie Mobile

Le système de téléphonie mobile modélisé par [Mil99] est le suivant :

— Il existe un réseau statique d'émetteurs qui sont tous connectés à un centre de contrôle.

— Chaque téléphone mobile est dans une voiture et connecté à un émetteur unique en utilisant une fréquence unique.

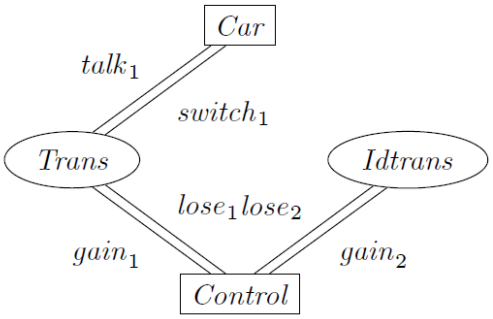
— Sur certains événements, par exemple évanouissement du signal, le téléphone mobile peut se connecter à un autre émetteur.

Un exemple simple d'un tel système est illustré dans la figure 3.11, où les auteurs ont montré comment on peut définir ce système sur BPL Tool.

```
val (switch1,  talk1,  lose1,  gain1) =
  ( "switch1","talk1","lose1","gain1")
val (switch2,  talk2,  lose2,  gain2) =
  ( "switch2","talk2","lose2","gain2")

val Car      = atomic ("Car"     -: 2)
val Trans    = atomic ("Trans"   -: 4)
val Idtrans  = atomic ("Idtrans" -: 2)
val Control  = atomic ("Control" -: 8)

val System1 =
      Car[talk1,switch1]
  `|` Trans[talk1,switch1,gain1,lose1]
  `|` Idtrans[gain2,lose2]
  `|` Control[lose1,talk2,switch2,gain2,
              lose2,talk1,switch1,gain1]
```



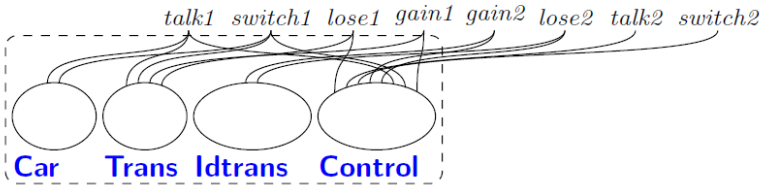


FIGURE 3.11 – Définition d'un système de téléphonie mobile

Le système se compose d'une voiture, deux émetteurs (un actif et l'autre passif) , et un centre de contrôle. Les contrôles sont atomiques (les nœuds ne doivent pas contenir d'autres nœuds)

3.6.2 Big Red

Big Red [FPH13] est un éditeur graphique (figure 3.12), développé sous Eclipse, permettant le développement des bigraphes et systèmes réactifs bigraphiques d'une manière visuelle.

Il établit un équilibre entre les deux aspects théorique et pratique des systèmes réactifs bigraphiques :

— Il offre un modèle bigraphique concret qui peut être étendu avec de nouvelles propriétés/contraintes pour implémenter d'autres versions des bigraphes.

— Le modèle bigraphique ne peut être modifié que par un script d'édition qui permet de regrouper les changements apportés au modèle et les valider ensemble.

— Il définit une représentation XML standard des objets du modèle (ainsi que des schémas pour les valider), qui peut être étendu pour inclure de façon portable des nouvelles propriétés et relations.

— Il est indépendant de la plateforme Eclipse, c'est-à-dire, des outils externes peuvent l'utiliser sans introduire beaucoup de dépendances.

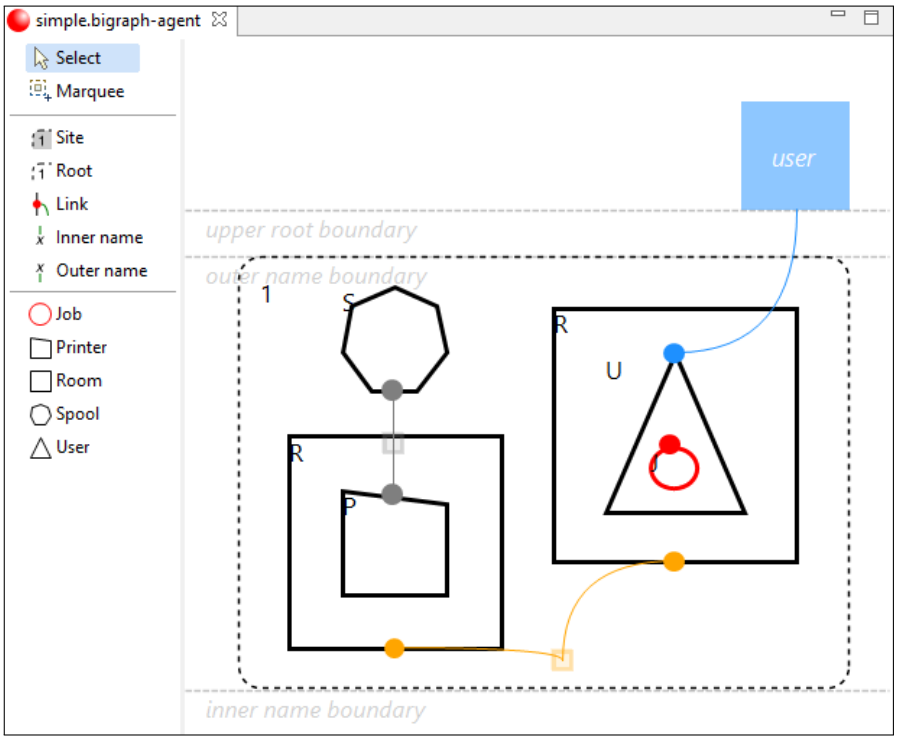


FIGURE 3.12 – Interface de l'éditeur Big Red [FPH13]

Implémentation

Big Red est implémenté comme un ensemble de plugins Eclipse étendant la plateforme Eclipse avec les formats de fichiers qui représentent les objets d'un système réactif bigraphique, les assistants de création des fichiers de modèles, et les éditeurs pour modifier ces modèles. Big Red définit plusieurs points d'extension, ceux-ci permettent de contribuer des extensions de Big Red et d'ajouter du support pour les nouveaux outils externe. En effet, l'implémentation du modèle bigraphique consiste en deux plugins :

— **Le modèle bigraphique** lui-même (et toute son infrastructure) qu'il n'est pas un plugin Eclipse, c'est-à-dire, il n'a pas de dépendances sur Eclipse, il n'apporte aucune extension et il peut également être utilisé à l'extérieur de la plateforme Eclipse.

— **Le wrapper** consiste en un plugin Eclipse qui implémente les mécanismes d'extensibilité de la plateforme Eclipse et les expose comme des points d'extension.

Intégration

Big Red définit un point d'extension spéciale pour les plugins qui souhaitent opérer sur le modèle des systèmes réactifs bigraphiques, ce qui offre aux développeurs la possibilité d'écrire des sous-programmes qui travaillent avec les objets du modèle de Big Red, sans plonger trop profondément dans le fonctionnement de l'environnement Eclipse. Aussi, il s'interface facilement avec les outils de bigraphe existants, tels que le model-checker BigMC [PDH12] pour permettre l'exécution de modèles bigraphiques.

L'intégration entre Big Red et BigMC est implémentée comme un plugin qui convertit un modèle Big Red en sa représentation du langage de termes BigMC, ensuite, il exécute BigMC comme un sous-processus et analyse les résultats dans Big Red de sorte qu'ils peuvent être visualisés.

3.6.3 BigMC

BigMC (Bigraphical Model Checker) [PDH12] est un model-checker permettant la vérification formelle des systèmes réactifs bigraphiques. Il utilise une technique de matching pour explorer tous les états possibles de ces systèmes et vérifier si une propriété est vraie dans chaque état. L'un des principaux avantages de BigMC est la possibilité de fournir un contre-exemple ou une trace d'exécution dans le cas où la propriété n'est pas vérifiée. Les contre-exemples générés s'avèrent précieux pour découvrir les causalités ayant menées à des violations de propriétés.

Les principaux objectifs de BigMC sont les suivants :

— Un outil de vérification pour les modèles, les langages et les propriétés d'exactitude décrits comme des bigraphes, qui peut être instancié comme un vérificateur d'accessibilité ou d'un simulateur pour les différents langages dédiés aux bigraphes.

— Un outil tractable, approximation sûre de l'analyse d'interférence des règles de réaction, ce qui représente un progrès dans l'état de l'art de la théorie des bigraphes.

— Un nouveau mécanisme pour l'analyse de l'accessibilité dans les systèmes réactifs bigraphiques basé sur l'expression des propriétés d'exactitude en tant qu'un matching.

— un outil offrant la possibilité de vérifier les propriétés de sûreté de certains systèmes de transitions où les états sont infinis.

Langage de Termes

La grammaire complète du langage de termes de BigMC est donnée par la table 3.4 :

Tableau 3.4 – Langage de termes BigMC

M ::= E; M | E;
E ::= %passive k : arity
E ::= %active k : arity
E ::= %rule N T → T
E ::= %property N P
E ::= T → T | T
T ::= K.T | T | T | T ‖ T | $n | K | nil
K ::= k[L] | k
L ::= N, L | N
N ::= [a – zA – Z][a – zA – Z0 – 9]∗ | –
P ::= matches(T)|terminal()|P&&P|P ‖ P|!P

M se réfère à un modèle bigraphique qui peut être composé à partir d'autres modèles M et/ou expressions E. Une expression E peut être un contrôle (k), une règle de réaction $T \rightarrow T$, ou une propriété P. Un contrôle k doit être défini par la déclaration de la signature $\%active$ ou $\%passive$ définissant l'arité (nombre de ports) du contrôle ainsi que s'il est actif ou passif. Tout terme de la forme $T \rightarrow T$ est considéré comme une règle de réaction, où le premier terme T représente le redex, tandis que le second représente le reactum. Un terme T peut représenter un nœud, un site ou une région. Mais aussi, il peut être une combinaison de tous ces éléments.

Les propriétés sont exprimées comme des combinaisons de deux prédicats : $matches$ et $terminal$. Le premier prédicat $matches()$ décrit un certain redex T qu'on doit trouver ou qu'on ne doit pas trouver $!matches(T)$ dans chaque état possible du système. Par exemple, on peut écrire une propriété : %properyno_a!matches(a.$0) ; qui précise qu'on ne doit jamais trouver une correspondance pour le redex a.$0, sinon une violation sera signalée. le prédicat $terminal()$ est vrai si et seulement s'il n'y a pas d'autres états atteignables par une étape de réaction de l'état actuelle. Les prédicats peuvent être combinés avec les opérateurs booléens : ET (&&), OU (‖) et SAUF (!) pour former des expressions de propriétés plus complexes.

Exemple : CCS

Nous introduisons un exemple qui représente l'un des objectifs à court terme énoncés par le formalisme des systèmes réactifs bigraphiques : l'analyse et la représentation des calculs de processus. Nous donnons un code simple d'un fragment fini d'un calcul des systèmes communicants (CCS) [Mil09], et démontrons que ce code nous permet de représenter facilement les termes CCS de telle manière qu'ils peuvent être exécutés directement.

Pour notre fragment choisi, les termes sont donnés sous la signature :

$$P ::= 0 \quad | \quad c.P \quad | \quad \overline{c}.P \quad | \quad P \mid P'$$

où 0 est le processus nul, c.P est le processus qui accepte l'entrée par le canal c, puis procède comme P, $\overline{c}$.P est le processus qui sort par le canal c et procède ensuite comme P, et P | P est la composition parallèle des deux processus. La traduction du fragment CCS en langage de termes BigMC est donnée comme suit :

$$\begin{aligned} [0] &\stackrel{def}{=} \mathsf{nil} \\ [c.P] &\stackrel{def}{=} \mathsf{recv}[c].[P] \\ [\overline{c}.P] &\stackrel{def}{=} \mathsf{send}[c].[P] \\ [P|P'] &\stackrel{def}{=} [P]|[P'] \end{aligned}$$

Pour un terme donné CCS comme $\overline{x}.\overline{y}.0|x.0|y.0$ on se retrouve avec le terme BigMC :

$$\mathsf{send}[x].\mathsf{send}[y]|\mathsf{recv}[x]|\mathsf{recv}[y]$$

Où send et recv sont passifs et d'arité 1.

La règle de réaction décrivant la synchronisation entre les deux processus est donnée comme suit :

$$\%\mathsf{rule} \quad \mathsf{ccs} \quad \mathsf{send}[c].\$0|\mathsf{recv}[c].\$1 \rightarrow \$0|\$1;$$

Pour ce qui BigMC rapporte la découverte de la séquence d'états :

send[x].send[y]|recv[x]|recv[y]
send[y]|recv[y]
nil

3.6.4 BigraphER

BigraphER [Sev12] est un outil qui consiste en une bibliothèque OCaml et un outil en ligne de commande qui permettent la manipulation, la visualisation et la simulation efficaces des systèmes réactifs bigraphiques stochastiques. BigraphER peut être téléchargé sur le site http://dcs.gla.ac.uk/~michele/bigrapher.html.

Nous commençons par décrire l'architecture de l'outil en ligne de commande (figure 3.13). L'outil est composé de trois modules distincts : le compilateur (*compiler*), le moteur de matching (*matching engine*) et le moteur de réécriture (*rewriting engine*). Leurs interconnexions sont présentées dans la figure 3.13. L'entrée de l'outil est un fichier source contenant la spécification du modèle. Le langage utilisé, appelé BSL (*BigraphER Specification Language*), ressemble à la forme algébrique de bigraphes introduite dans la section 3.5. Les modules sont représentés par les rectangles dans la boîte en pointillé, alors que les flèches non étiquetées montrent la relation de dépendance entre ces modules.

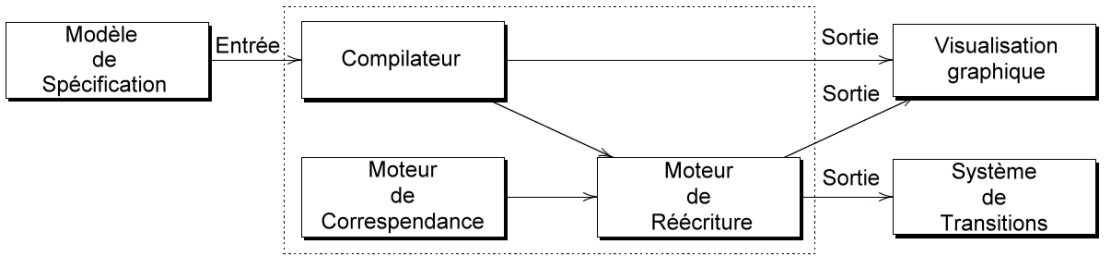


FIGURE 3.13 – Architecture de l'outil en ligne de commande [Sev12]

Le langage de spécification BigraphER permet de définir la signature du modèle, un ensemble de bigraphes et un ensemble de règles de réaction.

Un aperçu des principales caractéristiques du langage est présenté dans ce qui suit.

Une signature est spécifiée par une séquence de déclarations de contrôle de la forme suivante :

$$\mathsf{ctrl} \quad ctrl\_name = int;$$

où $ctrl\_name$ est un identifiant sous forme de chaîne de caractères. Chaque déclaration introduit un nouveau contrôle et l'attribue une arité. L'arité est un nombre entier int.

Les variables qui représentent les bigraphes sont déclarées de la même façon :

$$\mathsf{big}[\mathsf{init}]\, name = big\_exp$$

où $name$ est l'identifiant, $init$ est l'argument optionnel de l'initialisation qui permet de

spécifier un bigraphe comme état initial du modèle. Les exemples de l'expression big_exp sont les côtés droits des déclarations big de la figure 3.14.

Les opérateurs sur les expressions de big_exp sont les suivantes :

$$\_*\_ \quad \_+\_ \quad \_\|\_ \quad \_|\_ \quad \_.\_ \qquad \text{share\_by\_in}$$

Elles correspondent aux expressions algébriques : composition, produit tensoriel, produit parallèle, fusion, imbrication et partage, respectivement.

La déclaration de règles de réaction a la forme suivante :

$$\text{react} \quad name = big\_exp \rightarrow big\_exp;$$

Enfin, les règles de réaction stochastiques sont déclarées comme suit :

$$\text{sreact} \quad name = big\_exp \rightarrow big\_exp@float;$$

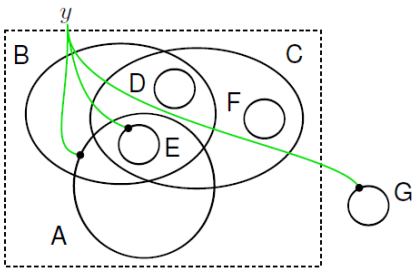
où $float$ indique le taux de réaction.

```
ctrl A = 1; ctrl E = 1; ctrl G = 1;
ctrl B = 0; ctrl C = 0; ctrl D = 0; ctrl F = 0;
big b = share f by phi in g;
big f = D.1 || E({y}).1 || F.1 || G({y}).1;
big phi = ([{1,2},{0,1,2},{2},{}], 3);
big g = A({y}) | B | C;
```

(a)

$B = \text{share } F \text{ by } \phi \text{ in } G$
$F = \mathsf{D}.1 \parallel \mathsf{E}_y.1 \parallel \mathsf{F}.1 \parallel \mathsf{G}_y.1$
$\phi = [\{1,2\},\{0,1,2\},\{2\},\emptyset]$
$G = \mathsf{A}_y \mid \mathsf{B} \mid \mathsf{C}$

(b)



(c)

FIGURE 3.14 – Comparaison entre une représentation dans le langage de spécification BigraphER (a) et la forme algébrique (b) du bigraphe $B : \epsilon \rightarrow \langle 1, y \rangle$ (c)

Compilateur

Le compilateur est le composant qui traduit un fichier source d'entrée en une représentation du modèle au moment de l'exécution. Chaque déclaration spécifie la liaison d'un identifiant à un type de données représentant soit un contrôle, un bigraphe, une règle de réaction ou une règle de réaction stochastique. Les contrôles et les règles de réaction sont stockés comme des entiers et de registres OCaml, respectivement.

Le codage d'un bigraphe est un peu plus complexe et nécessite deux structures de données spécialisées, une pour le graphe de places et l'autre pour le graphe de liens. Le compilateur est utilisé par le moteur de réécriture pour récupérer des structures de données correspondant à l'état initial et aux règles de réaction du modèle.

En outre, le compilateur peut également générer une représentation graphique de chaque bigraphe spécifié dans le fichier d'entrée à l'aide du générateur automatique de graphes Graphviz [EGK+02].

Moteur de Matching

Le moteur de matching implémente l'algorithme de matching pour les bigraphes avec partage introduits dans [BDGM07]. Il est utilisé par le moteur de réécriture pour appliquer des règles de réaction à un état et pour vérifier l'égalité des états.

L'implémentation est basée sur un codage SAT [SUC10] de l'algorithme de matching. Les solutions sont obtenues en passant l'instance SAT résultante du codage au solveur MiniSat [ES12].

Moteur de Réécriture

Le moteur de réécriture est le composant du système BigraphER qui calcule l'évolution dynamique des bigraphes. Il construit un graphe représentant le système de transition sous forme d'un processus de Markov à temps continu (*CTMC*) [Sev12] correspondant à un système réactif bigraphique stochastique.

Le moteur de réécriture est construit en appliquant des règles de réaction, de manière itérative à chaque état et ensuite, il mémorise les états résultants, jusqu'à un point fixe est atteint. Cela se produit lorsque tous les bigraphes obtenus par l'application des règles de réaction sont déjà présents dans le graphe.

Notez que le modèle est supposé avoir un espace d'état fini. Par conséquent, aucune commande n'est effectuée pour s'assurer que la boucle termine. Afin d'éviter un arrêt brutal

lorsque le programme court de mémoire, un commutateur précisant le nombre d'itérations peut être utilisé lorsque l'outil est lancé. Le moteur de réécriture peut produire soit un texte ou une représentation graphique du graphe. Il peut également donner une représentation graphique de chaque état.

La bibliothèque BigraphER OCaml fournit des interfaces de programmation pour les structures de données utilisées par l'outil en ligne de commande. Elle est équipée de fonctions pour calculer la composition de deux bigraphes et le résultat d'une application des règles de réaction.

Visualisation

Une représentation graphique des bigraphes peut être générée automatiquement par l'outil BigraphER (figure 3.15). l'implémentation de la fonction de visualisation utilise Graphviz, un logiciel de visualisation graphique open-source. De manière plus détaillée, le compilateur calcule une description textuelle d'un bigraphe puis, il l'envoie à l'outil de points qui génère automatiquement la disposition du graphe.

Différents modèles sont utilisés pour désigner les composants d'un bigraphe, par exemple, les sites et les racines sont représentés par des rectangles avec un conteur en pointillé et les liens sont dessinés comme des lignes vertes. Les contraintes de rang sur les racines et les noms externes les forcent à apparaitre en haut du diagramme. De même, les sites et les noms internes sont forcés vers le bas. Chaque hyperarc du graphe de liens est représenté par un nœud invisible auquel les ports et les noms externes sont liés.

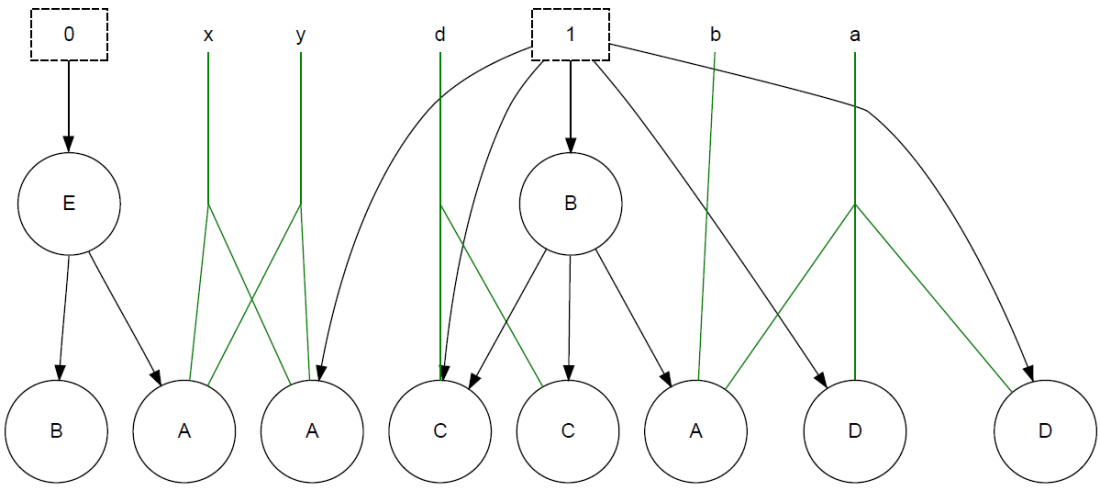


FIGURE 3.15 – Exemple d'une visualisation générée automatiquement

3.7 Conclusion

Dans ce chapitre, nous avons introduit la théorie des systèmes réactifs bigraphiques. Pour ce faire, nous avons d'abord commencé par décrire l'anatomie des bigraphes et leurs caractéristiques. Nous avons également exposé à travers des exemples les principales opérations que nous pouvons effectuer sur les bigraphes. Ensuite, nous avons introduit la dynamique des systèmes réactifs bigraphiques ainsi que leur langage de termes algébriques. Enfin, nous avons présenté les principaux outils de modélisation et simulation de ces systèmes.

Dans le chapitre suivant, nous abordons une approche de modélisation des systèmes sensibles au contexte basée sur les systèmes réactifs bigraphiques.

4 BigCAS : Modèle à base des BRS pour la Spécification des Systèmes Sensibles au Contexte

Sommaire

4.1 Introduction

Un système sensible au contexte est un système qui réagit par un comportement convenable en considérant les informations de contexte de son environnement. L'adaptation au contexte est une réaction à l'aspect dynamique de l'environnement. Par conséquent, l'environnement joue un rôle important pour la sensibilité au contexte. Cependant, les systèmes sensibles au contexte présentent de nouveaux défis dûs à l'impact des nouvelles technologies. Ces défis sont résumés en ce que nous appelons des besoins ubiquitaires ou encore des caractéristiques ubiquitaires, à savoir la sensibilité au contexte, la dynamicité, la distribution et la mobilité.

Nous avons vu dans le chapitre 2 que de nombreuses approches ont été proposées dans la littérature afin de modéliser les systèmes sensibles au contexte. Ces approches tournent principalement autour de techniques basées sur les modèles graphiques, de modèles basés sur les ontologies, de modèles basés sur des algèbres de processus, et de modèles orientés systèmes réactifs bigraphiques. Cependant, après avoir analysé ces différentes approches, nous avons conclu que les systèmes réactifs bigraphiques sont les mieux adaptés pour décrire des systèmes répondant aux différents critères d'ubiquité [CB14].

Dans ce chapitre, nous présentons un modèle formel basé sur les systèmes réactifs bigraphiques pour la modélisation de la structure et le comportement dynamique des systèmes sensibles au contexte. Tout d'abord, nous présentons le principe fondamental de l'approche proposée en introduisant les définitions formelles et les représentations bigraphiques associées aux différents aspects d'un système sensible au contexte. Puis, nous illustrons notre approche à travers une étude de cas d'un système d'éclairage dans une maison intelligente.

4.2 Principe de Modélisation

La modélisation est une étape fondamentale dans la conception des systèmes sensibles au contexte. Elle consiste à définir un modèle structurellement abstrait, et sémantiquement riche permettant de modéliser les concepts d'un système, les relations entre ces concepts et des contraintes sur eux.

Dans ce contexte, nous proposons un modèle formel à base des systèmes réactifs bigraphiques, dit : BigCAS (*Bigraphical Contexte-Aware System* en anglais) [CBB14b], dans lequel les principaux éléments d'un système sensible au contexte trouvent leur définition en termes des concepts bigraphiques tout en préservant leur sémantique. Pour cela, des correspondances entre les concepts des systèmes sensibles au contexte et ceux des systèmes réactifs bigraphique sont établies.

La description ci-après résume ces correspondances :

— La partie non-sensible au contexte est représentée par un bigraphe où les nœuds correspondants aux différentes entités internes.

— La partie sensible au contexte est modélisée par un bigraphe contenant des nœuds correspondants aux entités qui peuvent être déterminées ou influencées par le contexte.

— Chaque entité d'un système sensible au contexte est modélisée par un nœud et associée à un contrôle pour représenter son identité.

— L'aspect hiérarchique dans un système sensible au contexte est pris en charge par le concept d'imbrication de nœuds.

— Les interactions entre les différentes entités d'un système sensible au contexte sont définies par des hyperarcs.

— Le changement de comportement d'un système sensible au contexte est modélisé en tant que résultat de l'application d'une nouvelle catégorie des règles de réaction qui prennent en charge la reconfiguration interne et contextuelle.

Le tableau 4.1 présente les correspondances entre les éléments du système sensible au contexte et ceux du système réactif bigraphique.

Tableau 4.1 – Sémantique bigraphique des éléments d'un système sensible au contexte

Élément	Sémantique bigraphique
Structure	
Partie sensible au contexte	Bigraphe : $C = (V_{C_{id}}, E_{C_{id}}, ctrl_{C_{id}}, G^P_{C_{id}}, G^L_{C_{id}}) : I \to K$
Partie non-sensible au contexte	Bigraphe : $S = (V_S, E_S, ctrl_S, G^P_S, G^L_S) : K \to J$
Système sensible au contexte	Bigraphe : $S_{C_{id}} : I \to J \stackrel{\text{def}}{=} S \circ C$
Entité	Nœud : $v_i \in V^S_{C_{id}} / V^S_{C_{id}} = V_S \uplus V_{C_{id}}$ et $V_S \cap V_{C_{id}} = \emptyset$
Identité	Contrôle : $K \in \mathcal{K}^S_{C_{id}} / \mathcal{K}^S_{C_{id}} = \mathcal{K}_S \uplus \mathcal{K}_{C_{id}}$
Interaction	Hyperarc : $e_i \in E^S_{C_{id}} / E^S_{C_{id}} = E_S \uplus E_{C_{id}}$ et $E_S \cap E_{C_{id}} = \emptyset$
Comportement	
Reconfiguration interne	Règle de réaction : $R_S = (R : z \to J, R' : z' \to J, \eta)$
Reconfiguration contextuelle	Règle de réaction : $R_C = (R : m \to K, R' : m' \to K, \eta)$
Changement de contexte	Règle de réaction composite : $S_{C_{id}} \xrightarrow{\mathscr{R}} S_{C_{id'}} / \mathscr{R} = (R_{id}, R'_{id'}, \eta)$

Structurellement, BigCAS offre une séparation claire entre la partie sensible au contexte et la partie non-sensible au contexte. Chaque partie du système est modélisée séparément

par un bigraphe distinct. La composition de ces bigraphes est également un bigraphe qui modélise la structure générale d'un système sensible au contexte. En outre, la dynamique de ces systèmes est prise en charge par des règles de réaction permettant de modéliser les reconfigurations internes et contextuelles du système.

Dans cette section, nous donnons les définitions formelles nécessaires à la compréhension de ce manuscrit, ainsi que quelques exemples illustratifs.

4.2.1 Aspect Structurel

La définition structurelle des systèmes réactifs bigraphiques est enrichie pour représenter la structure d'un système sensible au contexte. Pour ce faire, la partie sensible au contexte et la partie non-sensible au contexte du système sont modélisées, chacune séparément, en utilisant deux bigraphes distincts. $S : K \rightarrow J$ est le bigraphe qui modélise la partie non-sensible au contexte, et $C_{id} : I \rightarrow K$ est le bigraphe modélisant la partie sensible au contexte. Puis, les deux bigraphes sont fusionnés en appliquant l'opération de composition (c'est-à-dire $S \circ C_{id}$) afin de représenter l'ensemble du système défini par $S_{C_{id}} : I \rightarrow J$. Enfin, les éléments non-sensibles au contexte et les éléments sensibles au contexte sont appelés respectivement *internes* et *contextuelles*.

Structure non-Sensible au Contexte

La partie non-sensible au contexte est la partie du système dont la structure reste constante vis-à-vis les changements de contexte. Généralement, elle consiste en l'ensemble des entités initiales qui composent le système, indépendamment de tout contexte.

D'un point de vue bigraphique, chaque entité initiale est modélisée par un *nœud interne* et les interactions entre elles sont modélisées par des *hyperarcs internes*.

La définition formelle du bigraphe de la partie non-sensible au contexte est donnée ci-dessous :

Définition 4.1 (Bigraphe S)**.** Un bigraphe S modélisant la partie non-sensible au contexte d'un système est donné par :

$$S = (V_S, E_S, ctrl_S, G_S^P, G_S^L) : K \rightarrow J$$

— V_S est un ensemble fini de nœuds où chaque nœud $v_i \in V_S$ est dit : *nœud interne* modélisant une entité non-sensibles au contexte.

— E_S est un ensemble fini d'hyperarcs où chaque hyperarc $e_i \in E_S$ est dit : *hyperarc interne*.

— $ctrl_S : V_S \rightarrow \mathcal{K}_S$ est une transformation qui associe à chaque nœud interne $v_i \in V_S$ un

contrôle $k \in \mathcal{K}_S$ indiquant le nombre de ports fixes et son comportement dynamique.

— $G_S^P = (V_S, prnt_S, ctrl_S) : k \to n$ est le graphe de places associé à S, où $prnt_S : m \uplus V_S \to V_S \uplus n$ est une fonction de parenté qui associe à chaque nœud ou site interne son parent hiérarchique.

— $G_S^L = (V_S, E_S, ctrl_S, link_S) : Z \to Y$ est le graphe de liens de S, où $Link_S : P \uplus Z \to E \uplus Y$ est une transformation montrant le flux de données des ports P ou les noms internes Z vers les hyperarcs E ou les noms externes Y, tel que l'ensemble de ports P est défini par $P \stackrel{\text{def}}{=} \{(v,i) | v \in V_S \wedge i \in ar(ctrl(v))\}$, c'est-à-dire, un port est représenté par une paire de nœud $v \in V_S$ et un indice i.

— $K = \langle k, Z \rangle$ et $J = \langle n, Y \rangle$ représentent respectivement les interfaces internes et externes du bigraphe S, où k est le nombre de sites, Z est l'ensemble des noms internes, n est le nombre de régions, et Y est l'ensemble des noms externes.

Exemple 4.1 (Bigraphe S)**.** Soit un bigraphe S (figure 4.1) modélisant la partie non-sensible au contexte d'un système, donné par $S : \langle 3, \{x, y\} \rangle \to \langle 2, \emptyset \rangle$ tel que :

— $V_S = \{v_0, v_2, v_4\}$ représente l'ensemble des nœuds internes.

— $E_S = \emptyset$.

— $ctrl_S = \{v_0 : 1, v_2 : 0, v_4 : 2\}$ indique le nombre de ports fixes de chaque nœud. Par exemple, le nœud v_0 a un seul port fixe.

— $prnt^S_{C_{id}} = \{v_0 : \emptyset, v_2 : v_0, v_4 : \emptyset\}$ indique le parent hiérarchique de chaque nœud interne. Par exemple, v_0 est le parent hiérarchique de v_2.

— $K = \langle 3, \{x, y\} \rangle$ est l'interface interne du bigraphe S. $k = 3$ représente le nombre de sites où trois régions contenant des nœuds contextuels peuvent être hébergées, et $Z = \{x, y\}$ représente l'ensemble des noms internes.

— $J = \langle 2, \emptyset \rangle$ est l'interface externe du bigraphe S où $n = 2 \quad$ et $\quad Y = \emptyset$ représente le nombre de régions et l'ensemble des noms externes, respectivement.

Les figures 4.2 et 4.3 représentent, respectivement, le graphe de places $G_S^P : 3 \to 2$ et le graphe de liens $G_S^L : \{x, y\} \to \emptyset$ associés au bigraphe non-sensible au contexte S.

Structure Sensible au Contexte

La partie sensible au contexte est la partie du système dont la structure supporte certaine variabilité en fonction du contexte qui entoure le système. Le bigraphe formalisant cette partie permet de modéliser les informations de contexte pertinentes pour l'exécution du système et les interactions avec ses utilisateurs.

Formellement, les entités de contexte sont représentées par des *nœuds contextuels* et les relations entre ces entités par des *hyperarcs contextuels*.

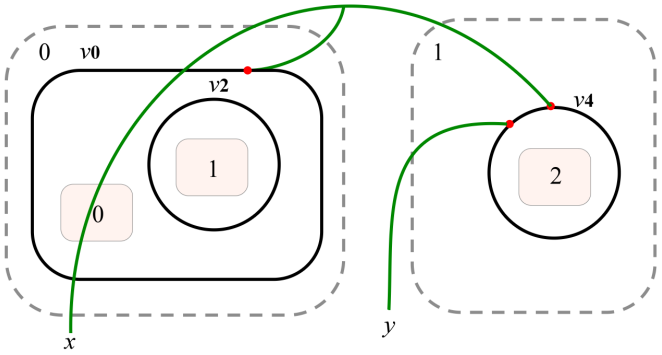


FIGURE 4.1 – Modélisation de la partie non-sensible au contexte

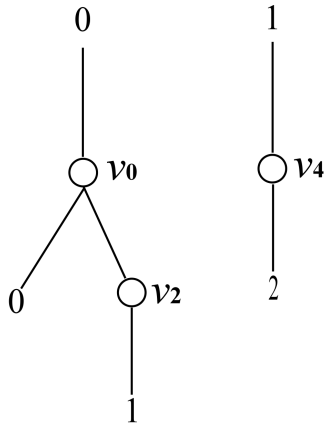


FIGURE 4.2 – Graphe de places de la partie non-sensible au contexte

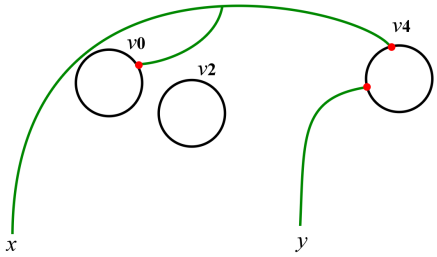


FIGURE 4.3 – Graphe de liens de la partie non-sensible au contexte

La définition formelle du bigraphe de la partie sensible au contexte est donnée ci-dessous :

Définition 4.2 (Bigraphe C_{id})**.** Un bigraphe C_{id} modélisant la partie sensible au contexte d'un système dans un contexte id est défini par :

$$C_{id} = (V_{C_{id}}, E_{C_{id}}, ctrl_{C_{id}}, G^P_{C_{id}}, G^L_{C_{id}}) : I \rightarrow K$$

— $V_{C_{id}}$ est un ensemble fini de nœuds où chaque nœud $v_i \in V_{C_{id}}$ est un nœud contextuel

modélisant une entité de contexte que ce soit une personne, une place ou un objet.

— $E_{C_{id}}$ est un ensemble fini d'hyperarcs où chaque hyperarc $e_i \in E_{C_{id}}$ représente l'interaction entre les entités de contexte qui les relie.

— $ctrl_{C_{id}} : V_{C_{id}} \to \mathcal{K}_{C_{id}}$ est une transformation qui associe à chaque nœud contextuel $v_i \in V_{C_{id}}$ un contrôle $k \in \mathcal{K}_{C_{id}}$ indiquant le nombre de ports contextuels de ce nœud. Un nœud contextuel est un nœud atomique.

— $G^P_{C_{id}} = (V_{C_{id}}, prnt_{C_{id}}, ctrl_{C_{id}}) : m \to k$ est le graphe de places associé à C_{id}, où $prnt_{C_{id}} : m \uplus V_{C_{id}} \to V_{C_{id}} \uplus n = \emptyset$. Un nœud contextuel est un nœud qui n'a pas de parent hiérarchique.

— $G^L_{C_{id}} = (V_{C_{id}}, E_{C_{id}}, ctrl_{C_{id}}, link_{C_{id}}) : X \to Z$ est le graphe de liens de C_{id}, où $Link_{C_{id}} : P_{C_{id}} \uplus X \to E_{C_{id}} \uplus Z$ est une transformation montrant le flux de données/événements des ports contextuels $P_{C_{id}}$ ou les noms internes X vers les hyperarcs contextuels $E_{C_{id}}$ ou les noms externes Z. Ce flux décrit d'une façon claire et visuelle les interactions entre les entités de contexte au sein d'un même système.

— $I = \langle m, X \rangle$ et $K = \langle k, Z \rangle$ représentent respectivement les interfaces internes et externes du bigraphe C_{id}, où m est le nombre de sites, X est l'ensemble des noms internes, k est le nombre de régions, et Z est l'ensemble des noms externes.

Exemple 4.2 (Bigraphe C_{id})**.** Soit un bigraphe C_{id} (figure 4.4) modélisant la partie sensible au contexte d'un système, donné par $C_{id} : \epsilon \to \langle 3, \{x, y\} \rangle$ tel que :

— $V_{C_{id}} = \{v_1, v_3, v_5\}$ représente l'ensemble des nœuds contextuels.

— $E_{C_{id}} = \{e_1\}$ représente l'ensemble des hyperarcs contextuels.

— $ctrl_{C_{id}} = \{v_1 : 2, v_3 : 2, v_5 : 1\}$ indique le nombre de ports contextuels de chaque nœud. Par exemple, le nœud v_1 a deux ports contextuels.

— $I = \langle 0, \emptyset \rangle = \epsilon$ est l'origine de C_{id}.

— $K = \langle 3, \{x, y\} \rangle$ est l'interface externe de C_{id} où $k = 3$ indique que trois nœuds contextuels (v_1, v_3, v_5) contenus dans trois régions $(0, 1, 2)$, respectivement, nécessitent une opération d'hébergement et que ces nœuds exigent des sites différents $(0, 1, 2)$. $Z = \{x, y\}$ est l'ensemble des noms externes.

Le graphe de places $G^P_{C_{id}} : 0 \to 3$ et le graphe de liens $G^L_{C_{id}} : \emptyset \to \{x, y\}$ sont donnés par les figure 4.5 et 4.6 respectivement.

Structure Composite

La définition formelle d'un bigraphe associé à un système sensible au contexte est donnée ci-dessous :

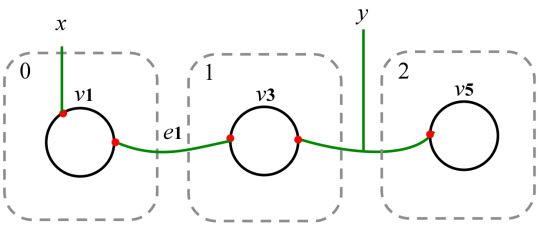


FIGURE 4.4 – Modélisation de la partie sensible au contexte

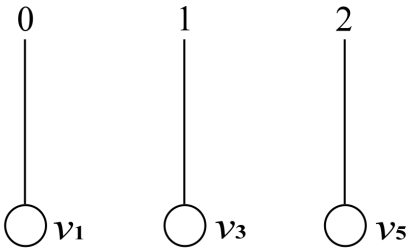


FIGURE 4.5 – Graphe de places de la partie sensible au contexte $G^{P}_{C_{id}} : 0 \rightarrow 3$

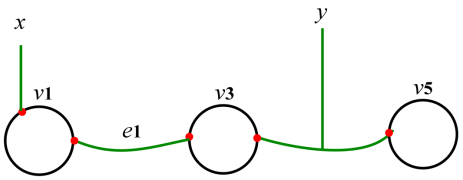


FIGURE 4.6 – Graphe de liens de la partie sensible au contexte $G^{L}_{C_{id}} : \emptyset \rightarrow \{x, y\}$

Définition 4.3 (Bigraphe $S_{C_{id}}$)**.** Un bigraphe $S_{C_{id}}$ modélisant un système sensible au contexte, dans un contexte id est défini par :

$$S_{C_{id}} \stackrel{\text{def}}{=} S \circ C_{id}$$

où

$$S_{C_{id}} = (V^{S}_{C_{id}}, E^{S}_{C_{id}}, ctrl^{S}_{C_{id}}, GP^{S}_{C_{id}}, GL^{S}_{C_{id}}) : I \rightarrow J$$

— $V^{S}_{C_{id}} = V_S \uplus V_{C_{id}}$ et $V_S \cap V_{C_{id}} = \emptyset$ est un ensemble fini de nœuds dans un contexte C_{id} donné par l'union disjointe de l'ensemble des nœuds système V_S et l'ensemble des nœuds contextuels $V_{C_{id}}$.

— $E^{S}_{C_{id}} = E_S \uplus E_{C_{id}}$ et $E_S \cap E_{C_{id}} = \emptyset$ est un ensemble fini d'hyperarcs dans un contexte C_{id} donné par l'union disjointe de l'ensemble des hyperarcs système E_S et l'ensemble des hyperarcs contextuels $E_{C_{id}}$.

— $\mathcal{K}^{S}_{C_{id}} = \mathcal{K}_S \uplus \mathcal{K}_{C_{id}}$ est une signature étendue, définie par un ensemble des contrôles où $ctrl^{S}_{C_{id}} : V^{S}_{C_{id}} \rightarrow \mathcal{K}^{S}_{C_{id}}$ est une nouvelle transformation qui associe à chaque nœud

$v_i \in V^S_{C_{id}}$ un contrôle $k \in \mathcal{K}^S_{C_{id}}$ indiquant le nombre de ses ports fixes et contextuels.

— $GP^S_{C_{id}} = G^P_S \circ G^P_{C_{id}}$ est le graphe de places associé à $S_{C_{id}}$ donné par la composition des graphes de places $G^P_S : k \to n$ et $G^P_{C_{id}} : m \to k$, où sa fonction de parenté $prnt^S_{C_{id}}$ est définie par : Si $w \in k \uplus V^S_{C_{id}}$ est un site ou un nœud dans $S_{C_{id}}$ alors

$$prnt^S_{C_{id}}(w) \stackrel{\text{def}}{=} \begin{cases} prnt_{C_{id}}(w) & \text{si } w \in k \uplus V_{C_{id}} \text{ et } prnt_{C_{id}}(w) \in V_{C_{id}}, \\ prnt_S(j) & \text{si } w \in k \uplus V_{C_{id}} \text{ et } prnt_{C_{id}}(w) = j \in m, \\ prnt_S(w) & \text{si } w \in V_S. \end{cases}$$

— $GL^S_{C_{id}} = G^L_S \circ G^S_{C_{id}}$ est le graphe de liens de $S_{C_{id}}$, donné par la composition des graphes de liens $G^L_S : Z \to Y$ et $G^L_{C_{id}} : X \to Z$, où $link^S_{C_{id}}$ est une transformation définie par : Si $q \in X \uplus P_S \uplus P_{C_{id}}$ est un point de $S \circ C$ alors

$$link(q) \stackrel{\text{def}}{=} \begin{cases} link_{C_{id}}(q) & \text{si } q \in X \uplus P_{C_{id}} \text{ et } link_{C_{id}}(q) \in E_{C_{id}}, \\ link_S(y) & \text{si } q \in X \uplus P_{C_{id}} \text{ et } link_{C_{id}}(w) = y \in Y, \\ link_S(q) & \text{si } q \in P_S. \end{cases}$$

.

— $I = \langle m, X \rangle$ et $J = \langle n, Y \rangle$ représentent respectivement les interfaces internes et externes du bigraphe $S_{C_{id}}$, où m est le nombre de sites, X est l'ensemble des noms internes, n est le nombre de régions, et Y est l'ensemble des noms externes.

Exemple 4.3 (Bigraphe $S_{C_{id}}$)**.** Soient un bigraphe non-sensible au contexte $S : \langle 3, \{x, y\} \rangle \to \langle 2, \emptyset \rangle$ (figure 4.1) et un bigraphe sensible au contexte $C_{id} : \epsilon \to \langle 3, \{x, y\} \rangle$ (figure 4.4). La composition des bigraphe S et C_{id} produit un nouveau bigraphe $S_{C_{id}} \stackrel{\text{def}}{=} S \circ C_{id}$ (figure 4.7) modélisant la structure du système sensible au contexte, donné par $S_{C_{id}} : \epsilon \to \langle 2, \emptyset \rangle$.

— $V^S_{C_{id}} = \{v_0, v_1, v_2, v_3, v_4, v_5\}$ représente l'ensemble des nœuds.

— $E^S_{C_{id}} = \{e_0, e_1, e_2\}$ représente l'ensemble des hyperarcs.

— $ctrl^S_{C_{id}} = \{v_0 : 1, v_1 : 2, v_2 : 0, v_3 : 2, v_4 : 2, v_5 : 1\}$ indique le nombre de ports fixes et contextuels de chaque nœud.

— $prnt^S_{C_{id}} = \{v_0 : \emptyset, v_1 : v_0, v_2 : v_0, v_3 : v_2, v_4 : \emptyset, v_5 : v_4\}$ indique que, après l'application de l'opération de composition, les nœuds contextuels v_1, v_3 et v_5 sont hébergés par les nœuds internes v_0, v_2 et v_3 respectivement.

— $I = \langle 0, \emptyset \rangle$ et $J = \langle 2, \emptyset \rangle$ représente, respectivement, l'interface interne et l'interface externe de $S_{C_{id}}$. $m = 0$ qui signifie que tous les sites sont occupés, c'est-à-dire, tous les nœuds contextuels sont hébergés, et $X = \emptyset$ et $Y = \emptyset$ indique que les noms internes de S et les noms externes de C_{id} sont fusionnés pour donner des nouveaux hyperarcs e_0 et e_2.

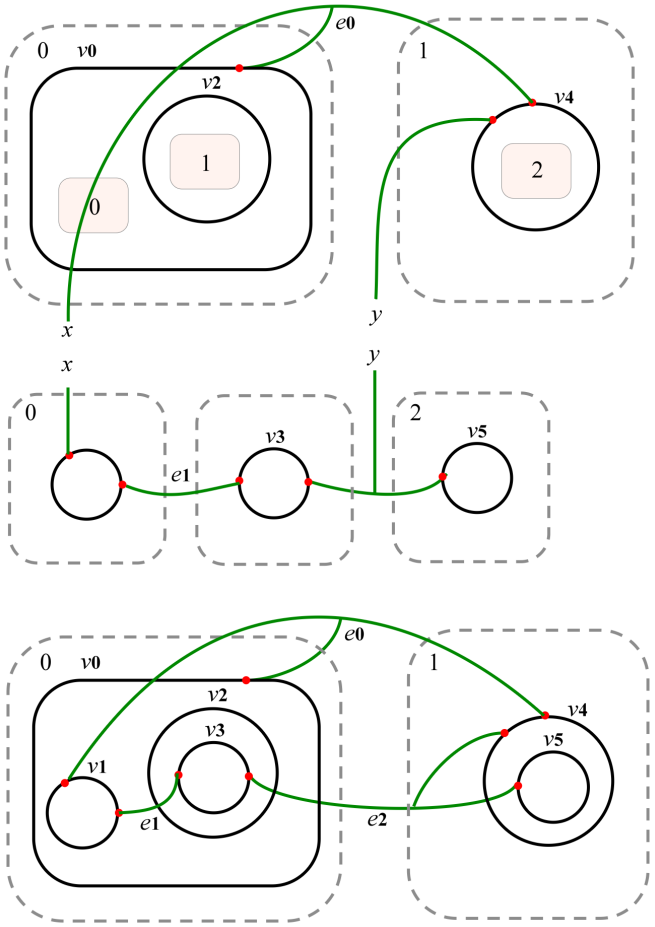


FIGURE 4.7 – Bigraphe composite d'un système sensible au contexte $S_{C_{id}} : \epsilon \rightarrow \langle 2, \emptyset \rangle$

Finalement, le graphe de places $GP^S_{C_{id}} : 0 \rightarrow 2$ (figure 4.8) est le résultat de la composition des graphes de places $G^P_S \circ G^P_{C_{id}}$, et le graphe de liens $GL^S_{C_{id}} : \emptyset \rightarrow \emptyset$ (figure 4.9) est le résultat de la composition des graphes de liens $G^L_S \circ G^L_{C_{id}}$.

4.2.2 Aspect Comportemental

Le comportement dynamique d'un système sensible au contexte est le résultat de transition d'un contexte à un autre, déclenchée par un événement qui affecte l'état du système. Formellement, une transition de contexte a la forme $S_{C_{id}} \xrightarrow{\mathscr{R}} S_{C_{id'}}$ et définie par une règle de réaction donnée par $\mathscr{R}$ où $S_{C_{id}}$ modélise l'état actuel d'un système sensible au contexte et $S'_{C_{id'}}$ modélise le prochain état du système dans un nouveau contexte. $\mathscr{R}$ est une règle de réaction dite *composite* représentant une séquence de règles de réaction qui se produisent dans chaque partie du système sensible au contexte pour le faire passer d'un état à un autre.

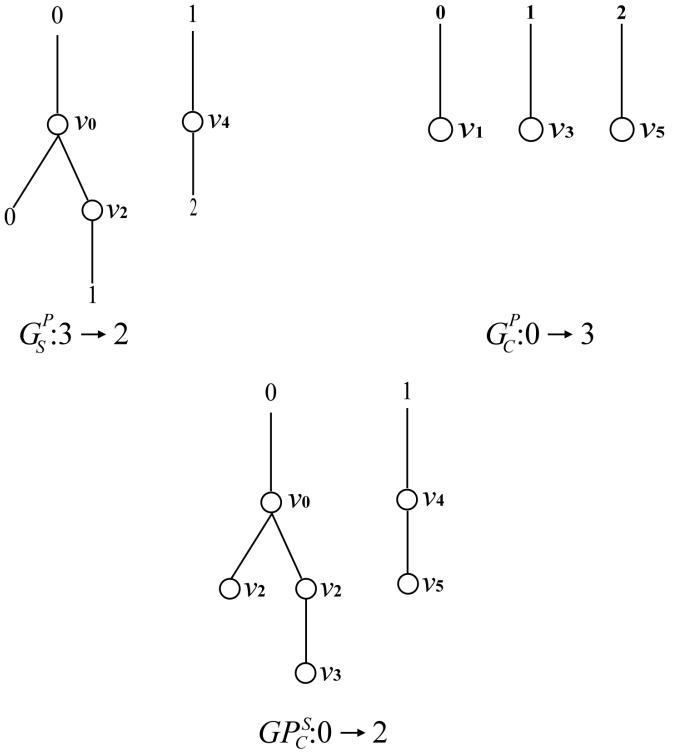


FIGURE 4.8 – Graphe de places composite

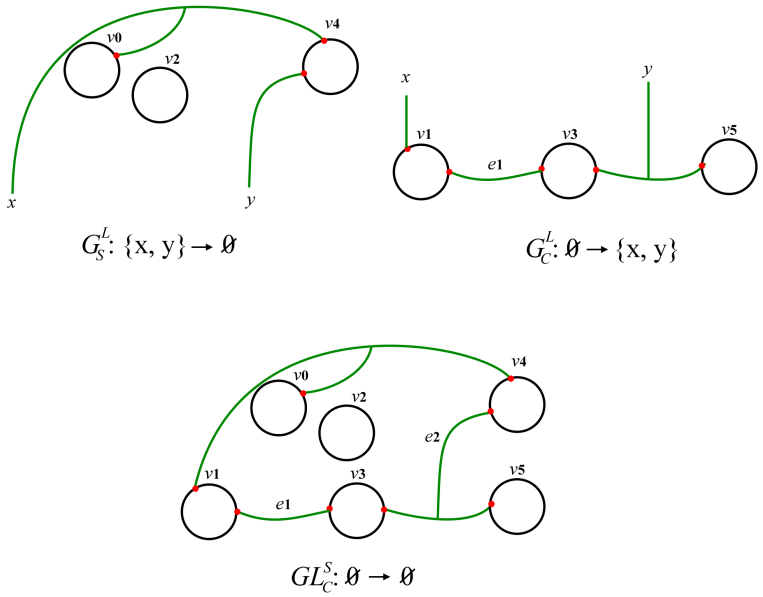


FIGURE 4.9 – Graphe de liens composite

Nous notons que chaque partie (c'est-à-dire la partie sensible au contexte et la partie non-sensible contexte) a ses propres règles de réaction, appelées *règles de réaction contextuelles* et

règles de réaction internes, respectivement. Cependant, ces règles de réaction sont effectuées indépendamment les unes des autres.

Reconfiguration Interne

Les règles de réaction internes sont des règles appliquées au bigraphe non-sensible au contexte pour définir le comportement dynamique interne d'un système sensible au contexte.

La définition formelle d'une règle de réaction interne est donnée ci-dessous.

Définition 4.4 (Règle de réaction interne)**.** Une règle de réaction interne est donnée par :

$$R_S = (R : z \to J, R' : z' \to J, \eta)$$

où $R : z \to J$ est un bigraphe appelé redex (le pattern à changer), $R' : z' \to J$ est également un bigraphe appelé reactum (le nouveau pattern) et l'instanciation $\eta : z' \to z$ est une fonction de correspondance entre z' et z.

Exemple 4.4. La figure 4.10 représente une règle de réaction interne appliquée au bigraphe non-sensible au contexte S (figure 4.1) dont la forme est :

$$R_S = (R : 3 \to \langle 2, \emptyset \rangle), R' : 2 \to \langle 2, \emptyset \rangle, 2 \to 3$$

L'application de la règle R_S a comme effet la suppression du nœud interne v_2, ce qui exprime la suppression de l'élément non-sensible au contexte modélisé par ce nœud ainsi que toutes ses connections.

L'expression algébrique correspondante à la règle de réaction interne R_S est comme suit :

$$x/v_0.(d_0 \mid v_2.d_1) \parallel y/v_4.d_2 \to x/v_0.d_0 \parallel y/v_4.d_2$$

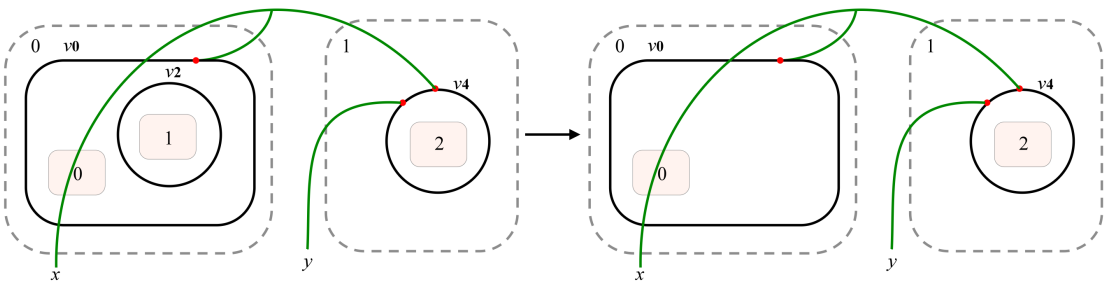


FIGURE 4.10 – Modélisation d'une règle de réaction interne

Reconfiguration Contextuelle

Les règles de réaction contextuelles sont des règles appliquées au bigraphe sensible au contexte pour modéliser la reconfiguration contextuelle d'un système sensible au contexte.

La définition formelle d'une règle de réaction contextuelle est donnée ci-dessous.

Définition 4.5 (Règle de réaction contextuelle)**.** Une règle de réaction contextuelle a la forme suivante :

$$R_C = (R : m \to K, R' : m' \to K, \eta)$$

où $R : m \to K$ est un bigraphe appelé redex dans un contexte id, $R' : m' \to K$ est également un bigraphe appelé reactum dans un nouveau contexte id' et l'instanciation $\eta : m' \to m$ est une fonction de correspondance entre m' et m.

Exemple 4.5. Soit R_C (figure 4.11) une règle de réaction contextuelle appliquée au bigraphe sensible au contexte C_{id} représenté dans la figure 4.4 et donnée par :

$$R_C = (R : 0 \to \langle 3, \{x, y\}\rangle, R' : 0 \to \langle 2, \{x, y\}\rangle, 0 \to 0)$$

La règle $R_{C_{id}}$ consiste en la suppression du nœud contextuel v_3 et toutes ses connexions, ce qui exprime la suppression de l'élément sensible au contexte modélisé par v_3 ainsi que toutes ses connections.

L'expression algébrique de la règle de réaction contextuelle $R_{C_{id}}$ est donnée ci-dessous.

$$x/v_1 \parallel v_3 \parallel y/v_5 \to x/v_1 \parallel y/v_5$$

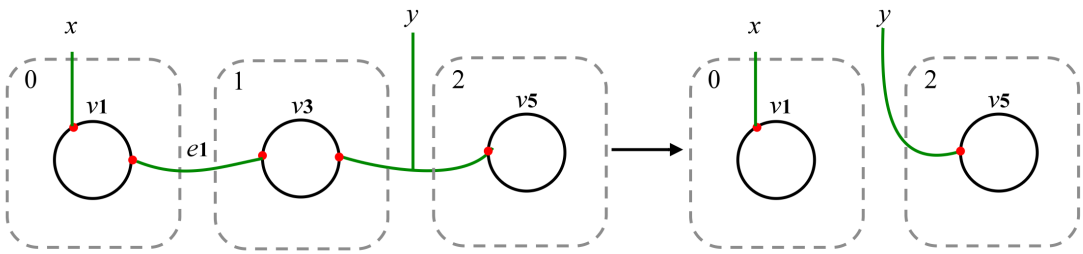


FIGURE 4.11 – Modélisation d'une règle de réaction contextuelle

Reconfiguration Composite

Une règle de réaction composite est le résultat de l'application d'une séquence finie des règles de réaction internes et contextuelles. En d'autres termes, elle représente les différents changements de comportement général d'un système sensible au contexte.

La définition formelle d'une règle de réaction composite est énoncée ci-dessous.

Définition 4.6 (Règle de réaction composite)**.** Une règle de réaction composite est un triplet de la forme :

$$\mathscr{R} = (R_{id} : m \to J, R'_{id'} : m' \to J, \eta)$$

où $R_id : m \to J$ est un bigraphe redex dans un contexte id, $R'_{id'} : m' \to J$ est un bigraphe reactum dans un nouveau contexte id' et l'instanciation $\eta : m' \to m$ est une transformation.

Exemple 4.6. La figure 4.12 représente le résultat de l'application de la règle de réaction interne R_S(figure 4.10) et la règle de réaction contextuelle R_C (figure 4.11)

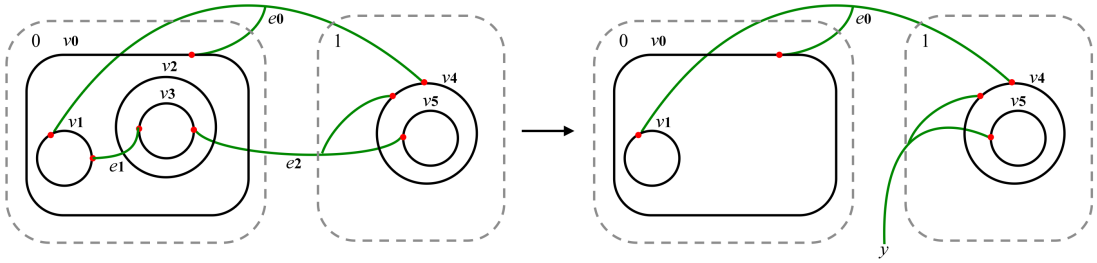


FIGURE 4.12 – Modélisation d'une règle de réaction composite

4.3 Étude de Cas : Maison Intelligente

La maison est un endroit particulièrement important pour tout le monde, il s'agit d'un lieu où l'on se sent en sécurité et où l'on aime se détendre. La plus part des gens, et plus particulièrement les personnes âgées, passent la majorité de leur temps à domicile, d'où l'influence particulière de l'habitat sur la qualité de vie de la personne. Par conséquent, l'amélioration des conditions de confort et de bien-être dans l'habitat apparaît donc comme une nécessité impérieuse dans notre vie domestique.

L'idée d'une maison automatisée associée au concept de domotique remonte aux années 1980, où des applications très diverses étaient inventoriées sur le confort et la sécurité des personnes et des biens. La domotique regroupe l'ensemble des technologies permettant l'automatisation des équipements d'un habitat tels que la lumière, le chauffage, la climatisation, le téléviseur, le portail d'entrée, la porte de garage, ou encore les volets roulants [Ald03]. Ce concept vise à apporter des solutions technologiques pour répondre aux besoins de confort des habitants, à savoir la sécurité, la gestion d'énergie, et le pilotage à distance des équipements. Cependant, la domotique, malgré des productions très élaborées et bien ciblées sur les équipements domestiques dès le début des années 1990, fut marqué par un échec commercial et le marché n'a jamais pu atteindre les objectifs espérés [Sar89]. En effet, plusieurs raisons

ont conduit à cet échec, dont les plus souvent invoquées sont des besoins et des usages mal identifiés, des technologies peu développées et une absence de normalisation.

Avec l'apparition de l'informatique ubiquitaire [Wei93, Wei94, Wei95] et les promesses déçues de la domotique dans les années 1990, les acteurs du domaine ont orienté la domotique vers la maison intelligente (*smart home* en anglais) pour effacer l'image technologique induite par ce terme. Une maison intelligente [Ald03, CEEC08] est une résidence équipée de technologie d'intelligence ambiante, qui anticipe et répond aux exigences de ses habitants en essayant de gérer de manière optimale leur confort et leur sécurité par action sur la maison, et en mettant en œuvre des connexions avec le monde extérieur. En effet, la maison intelligente n'est pas seulement un espace physique constitué d'un ensemble d'outils destinés à aider les habitants dans leurs tâches de la vie quotidienne, mais aussi un espace de vie socioculturel pour les humains où leur qualité de vie et la perception qu'ils ont, doivent être améliorées par la technologie et l'intelligence ambiante [FDCP+05].

Les solutions, prototypes et expérimentations traitant de la maison intelligente sont nombreuses et présentent une grande variété. Un aperçu de quelques projets réalisés est présenté ci-dessous.

— House_n [Int02]

— CASAS [CCTK13]

— The Intelligent Dormitory (iSpace) [HDPC02]

— GER'HOME [ZBT+09]

— Aging In Place [SAP+09]

— CompanionAble [BEH+09]

— SOPRANO (Service-Oriented Programmable Smart Environments for Older Europeans) [WSK08]

— SM4ALL (Smart Home For All) [CCdL+09]

En effet, le but des maisons intelligentes est de créer un environnement informatique transparent pour améliorer les conditions de vie des gens au quotidien, et les assister dans les situations diverses lorsqu'ils sont à leur domicile. Par exemple, afin de diminuer les gaspillages de ressources énergétiques, il est possible de mettre en veille les dispositifs de chauffage ou de climatisation quand les habitants sont absents ou adapter automatiquement l'utilisation des ressources électriques en fonction des besoins des habitants.

Au-delà de la gestion d'énergie, la maison intelligente offre divers services (voir figure 4.13 aux habitants tels que la sécurité, le pilotage des différents médias simultanément et à tout endroit de la maison (télévision, chaîne Hi-Fi, etc.), les soins de santé (*e-Health*), l'assistance à la vie quotidienne (ménage, surveillance, aide aux personnes handicapées, etc). Pour offrir ces différents services, la maison intelligente doit :

FIGURE 4.13 – Services de la maison intelligente

— **Capturer** - la maison intelligente acquiert des informations en provenance des différents équipements.

— **Comprendre** - la maison intelligente analyse les informations capturées, ce qui permet de comprendre le contexte dans lequel se trouvent la maison et ses habitants.

— **Agir** - la maison intelligente utilise ces informations pour fournir des services adéquats.

— **Apprendre**- la maison intelligente prend en compte les conséquences de ses interventions en vue des futures prises de décision.

Nous constatons donc que la maison intelligente repose sur un large panel d'informations contextuelles. Cette diversité est nécessaire si nous souhaitons mettre au point des services complexes.

L'éclairage automatique est un des services les plus communément retrouvés dans les maisons intelligentes. Dans ce qui suit, nous présentons le service d'éclairage automatique afin d'illustrer comment notre approche permet de modéliser les différentes informations contextuelles fournies par ce service.

4.3.1 Système d'Éclairage Intelligent

Le système d'éclairage intelligent (*Smart Lightning System* en anglais) est un élément fondamental de la maison intelligente, conçu principalement pour assurer un confort optimal des habitants, en faisant évoluer l'intensité lumineuse selon différents paramètres circonstanciels (luminosité naturelle, niveau d'occupation d'un lieu, mouvement, etc.). Cela permet de maximiser l'économie d'énergie et d'offrir la flexibilité et les personnalisations nécessaires pour assurer un confort optimal des habitants, tout en conservant une utilisation simplifiée. En outre, des recherches prouvent que des conditions d'éclairage optimales contribuent à l'accroissement des aptitudes cognitives, la diminution de la fatigue oculaire et l'amélioration de la satisfaction au travail.

Le système d'éclairage intelligent est le résultat de la combinaison de dispositifs d'éclairage performants (basés généralement sur la technologie LED), avec un système de gestion composé de différents dispositifs électroniques (capteurs de luminosité, présence, etc.), et d'une intelligence embarquée.

Ce type de système offre plusieurs fonctionnalités telles que :

— Éclairage allumé uniquement lorsqu'il est nécessaire, grâce à un capteur de présence (détection de présence, réglage en fonction des tâches).

— Niveau d'éclairage juste, grâce à un capteur de luminosité (régulation en fonction de la lumière du jour).

— Communication sans fil entre des luminaires installés en réseau.

— Pilotage intelligent de la lumière.

En effet, le système d'éclairage intelligent a initialement deux fonctionnalités : allumer et éteindre la lumière. Ces deux fonctionnalités ne doivent évidemment pas intervenir dans les mêmes situations. Afin de comprendre le fonctionnement de tel service, nous allons décrire un scénario dans lequel ce service peut être utile.

Scénario : Éclairage en Fonction de la Présence

L'éclairage en fonction de la présence consiste à allumer et éteindre la lumière en réponse à l'occupation d'une zone déterminée. Il ne dépend pas d'intervalles de temps ni de périodes programmées mais réagit plutôt à l'usage individuel d'un espace contrôlé. Le scénario ci-dessous illustre le fonctionnement du système d'éclairage en fonction de la présence, dans un couloir d'une maison intelligente.

1. Á l'état initial, l'éclairage est automatiquement éteint.
2. Lorsqu'un capteur détecte une personne, il allume sa lampe à un niveau de luminosité prédéterminé, et il communique simultanément l'information de la présence de la

personne avec ses lampes voisines, par un signal sans fil.

3. Á la réception de l'information, et même s'il ne peut détecter lui-même la personne, le capteur de la lampe voisine allume l'éclairage à un niveau spécifié (par exemple, moyen) et relaie simultanément un signal à ses propres lampes voisines, leur indiquant la proximité de la personne.

4. L'information se propage rapidement dans le couloir vers chaque capteur qui reçoit un signal indiquant à quel degré de proximité se trouve la personne, émettant un niveau de lumière préprogrammé basé sur cette information et l'ajustant ensuite en vue de créer la lumière suffisante.

5. Dès que la personne quitte le couloir, l'éclairage passe à l'état initial et toutes les lampes s'éteignent.

La figure 4.14 illustre le scénario de l'éclairage intelligent en fonction de la présence.

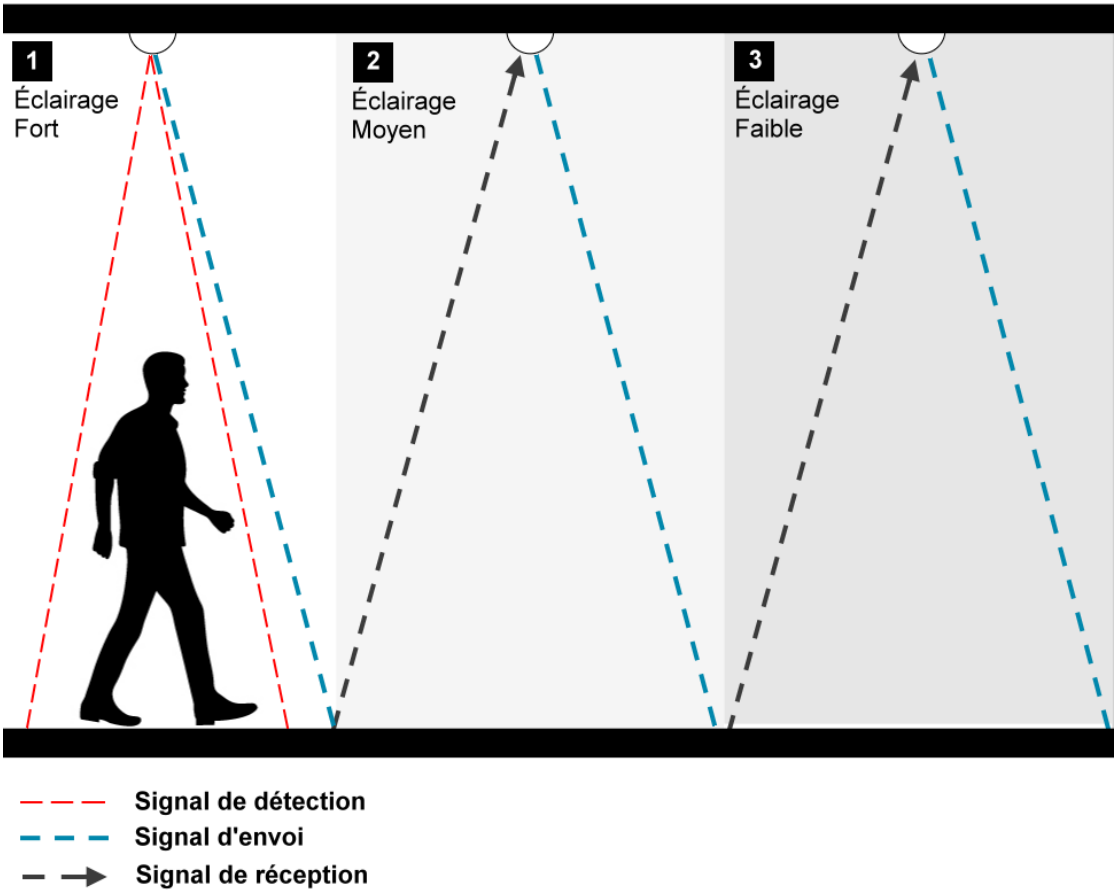


FIGURE 4.14 – Éclairage intelligent en fonction de la présence

4.3.2 Modélisation du Système d'Éclairage Intelligent

Dans cette section, nous détaillons l'application de notre approche sur un système d'éclairage intelligent (scénario 4.3.1 : éclairage en fonction de la présence).

La figure 4.15 représente le bigraphe formalisant l'état initial du système d'éclairage intelligent avant l'arrivée d'une personne. En effet, nous pouvons distinguer les principales entités d'un système d'éclairage intelligent représentées par des nœuds internes, qui sont : une chambre (chambre), une personne (personne), un couloir (couloir) et trois lampes (lampe_1, lampe_2 et lampe_3). Les liens ouverts x, y et z représentent les signaux de détection de mouvement. p est également un lien ouvert représentant l'information de la présence. En outre, il existe quatre modes d'éclairage : Éteint_i, Faible_i, Moyen_i et Fort_i représentent les informations contextuelles liées au système d'éclairage intelligent. Chaque information contextuelle est représentée par un nœud $mode_i$ où i indique le numéro de la lampe.

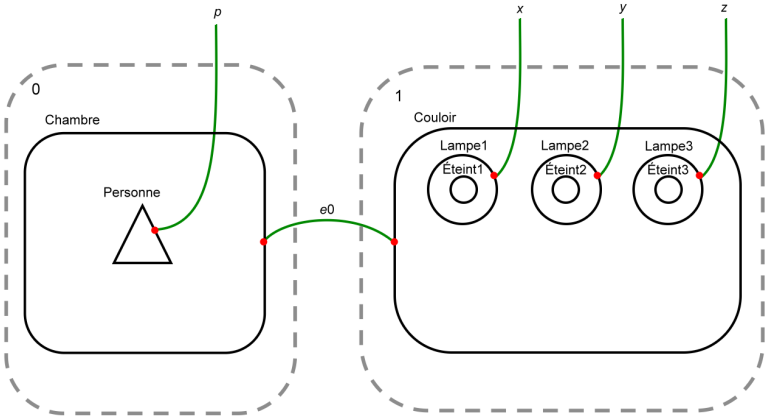


FIGURE 4.15 – Bigraphe de l'état initial du système d'éclairage intelligent

Le bigraphe formalisant le système d'éclairage intelligent à l'état initial est donné par :

$$S_{C_{initial}} : \epsilon \rightarrow \langle 2, \{p, x, y, z\} \rangle$$

où :

- $V^S_{C_{initial}} = \{Chambre, Personne, Couloir, Lampe_1, Lampe_2, Lampe_3, \text{É}teint_1, \text{É}teint_2, \text{É}teint_3\}$
- $E^S_{C_{initial}} = \{e_0\}$
- $GP^S_{C_{initial}} : 0 \rightarrow 2$
- $GL^S_{C_{initial}} : \emptyset \rightarrow \{p, x, y, z\}$
- $I = \langle 0, \emptyset \rangle$ et $J = \langle 2, \{p, x, y, z\} \rangle$ où $n = 2 \quad$ et $\quad Y = \{p, x, y, z\}$

L'évolution du système d'éclairage intelligent décrite dans le scénario 4.3.1 peut être formalisée par une séquence de règles de réaction, où chaque règle de réaction est déclenchée par un événement et modélise un état spécifique du système. La description algébrique de ces règles est donnée ci-dessous :

Événement (1) : Arrivée d'une personne.

$$R_S{:}Chambre.(p/Personne)\|Couloir.(x/Lampe_1.(\text{É}teint_1)|y/Lampe_2.(\text{É}teint_2)|z/Lampe_3.(\text{É}teint_3))$$

$$\rightarrow$$

$$Chambre\|Couloir.(Personne|x/Lampe_1.(d_1)|y/Lampe_2.(d_2)|z/Lampe_3.(d_3))$$

Événement (2) : Détection de la présence d'une personne.

$$R_C{:}Chambre\|Couloir.(Personne|x/Lampe_1.(d_1)|y/Lampe_2.(d_2)|z/Lampe_3.(d_3))$$

$$\rightarrow$$

$$Chambre\|Couloir.(Personne|Lampe_1.(Fort_1)|y/Lampe_2.(d_2)|z/Lampe_3.(d_3))$$

Événement (3) : Communication de l'information de la présence à les lampes voisines.

$$R_C{:}Chambre\|Couloir.(Personne|Lampe_1.(Fort_1)|y/Lampe_2.(d_2)|z/Lampe_3.(d_3))$$

$$\rightarrow$$

$$Chambre\|Couloir.(Personne|Lampe_1.(Fort_1)|Lampe_2.(Moyen_2)|z/Lampe_3.(d_3))$$

Événement (4) : Ajustement d'éclairage en fonction de proximité.

$$R_C{:}Chambre\|Couloir.(Personne|Lampe_1.(Fort_1)|Lampe_2.(Moyen_2)|z/Lampe_3.(d_3))$$

$$\rightarrow$$

$$Chambre\|Couloir.(Personne|Lampe_1.(Fort_1)|Lampe_2.(Moyen_2)|z/Lampe_3.(Faible_3))$$

Dans la description ci-dessus, nous remarquons clairement qu'à chaque fois un événement se produit, une lampe passe d'un mode vers un autre.

— Á l'arrivée de la personne, le système d'éclairage intelligent passe automatiquement à l'état prêt en initialisant le mode de chaque lampe (figure 4.1). L'initialisation est modélisée par les sites 1, 2 et 3 indiquant que les lampes sont prêtes à passer au mode de fonctionnement.

— Dès que la détection de la présence, la lampe_1 établit un lien e_1 avec la personne, et passe automatiquement du mode Eteint_1 au mode Fort_1.

— Ensuite, la lampe_1 communique l'information de la présence à la lampe_2 à travers le

lien e_2. Á la réception de l'information de présence, la $lampe_2$ passe du mode $Eteint_2$ au mode $Moyen_2$ et établit simultanément un lien e_2 avec la $lampe_3$ indiquant la proximité de la personne.

— Á la réception de l'information de proximité, $lampe_3$ bascule du mode $Eteint_3$ au mode $Faible_3$.

La figure 4.16 représente le bigraphe non-sensible au contexte à l'état de la présence d'une personne. Formellement, le bigraphe formalisant la partie non-sensible au contexte à l'état de

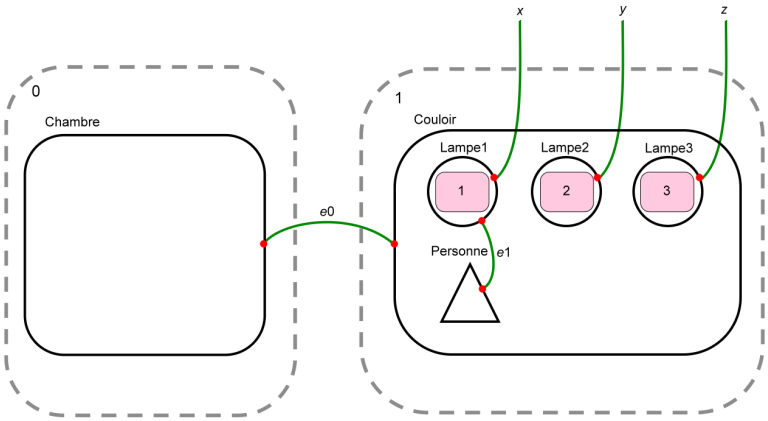


FIGURE 4.16 – Bigraphe S de l'état de la présence

la présence d'une personne est donné par :

$$S : \langle 3, \emptyset \rangle \rightarrow \langle 2, \{p, x, y, z\} \rangle$$

où :

— $V_S = \{Chambre, Personne, Couloir, Lampe_1, Lampe_2, Lampe_3\}$

— $E_S = \{e_0, e1\}$

— $G_S^P : 3 \rightarrow 2$

— $G_S^L : \emptyset \rightarrow \{p, x, y, z\}$

— $K = \langle 3, \emptyset \rangle$ où $k = 3$ et $Z = \emptyset$

— $J = \langle 2, \{p, x, y, z\} \rangle$ où $n = 2$ et $Y = \{p, x, y, z\}$

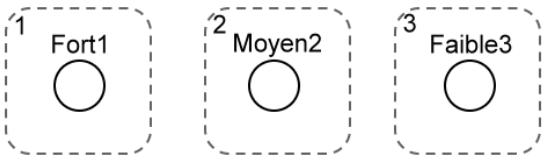


FIGURE 4.17 – Bigraphe $C_{présence}$ de l'état de la présence

Le bigraphe représenté dans la figure 4.17 modélise la partie sensible au contexte à l'état de la présence d'une personne et donné par :

$$C_{présence} : \epsilon \rightarrow \langle 3, \emptyset \rangle$$

où :

— $V_{C_{présence}} = \{Fort_1, Moyen_2, Faible_3\}$

— $E_{C_{présence}} = \emptyset$

— $G^P_{C_{présence}} : 0 \rightarrow 3$

— $G^L_{C_{présence}} : \emptyset \rightarrow \emptyset$

— $I = \epsilon$ et $K = \langle 3, \emptyset \rangle$ où $k = 3$ et $Z = \emptyset$

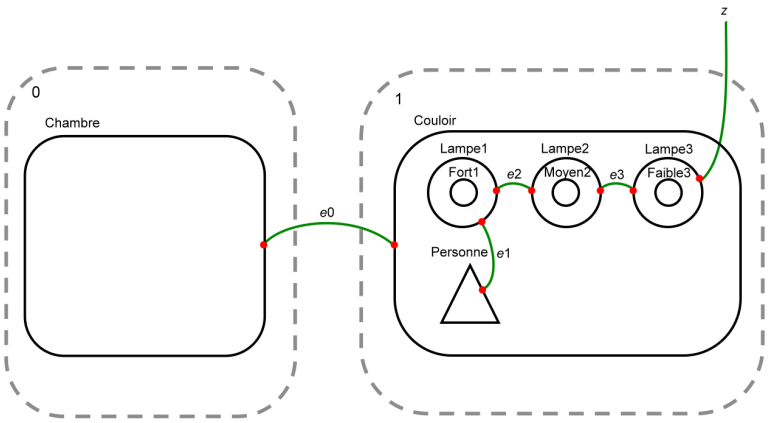


FIGURE 4.18 – Bigraphe de l'état de la présence d'une personne

La figure 4.18 représente le bigraphe $S_{C_{présence}}$ modélisant la structure générale du système d'éclairage intelligent à l'état de la présence d'une personne.

Formellement, le bigraphe $S_{C_{présence}}$ est le résultat de la composition $S \circ C_{présence}$ tel que :

$$S_{C_{présence}} : \epsilon \rightarrow \langle 2, \{z\} \rangle$$

où :

— $V^S_{C_{présence}} = \{Chambre, Personne, Couloir, Lampe_1, Lampe_2, Lampe_3, Fort_1, Moyen_2, Faible_3\}$

— $E^S_{C_{présence}} = \{e_0, e1, e2, e3\}$

— $GP^S_{C_{présence}} : 0 \rightarrow 2$

— $GL^S_{C_{présence}} : \emptyset \rightarrow \{z\}$

— $I = \epsilon$ et $J = \langle 2, \{z\} \rangle$ où $n = 2$ et $Y = \{z\}$

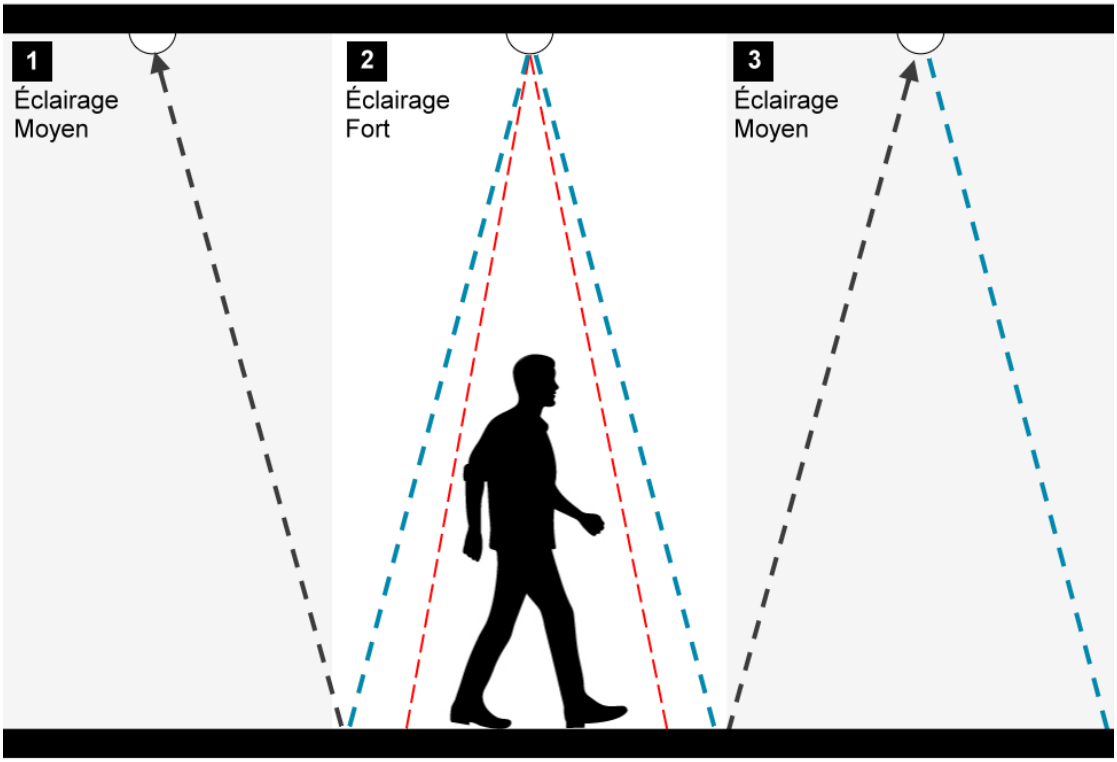


FIGURE 4.19 – Éclairage intelligent en fonction du mouvement

Événement (5) : Ajustement d'éclairage en fonction du mouvement.

$$Chambre \| Couloir.(Personne | Lampe_1.(Fort_1) | Lampe_2.(Moyen_2)) | z / Lampe_3.(d_3))$$

$$\rightarrow$$

$$Chambre \| Couloir.(Lampe_1.(Moyen_1) | Personne | Lampe_2.(Fort_2)) | z / Lampe_3.(Moyen_3)$$

Durant le passage d'une personne dans le couloir, le système d'éclairage adapte le mode de chaque lampe en fonction de ce mouvement. Par exemple, quand la lampe_2 détecte une personne (figure 4.20). La lampe_2 passe automatiquement au mode Fort_2 et envoi l'information de présence à ses proches voisins lampe_1 et lampe_3 qui passent aussi au mode Moyen.

Le bigraphe formalisant le système d'éclairage intelligent à l'état du mouvement d'une personne $S_{C_{mouvement}}$ (figure 4.22) est donné par la composition des bigraphes $S \circ C_{mouvement}$ représentés respectivement par les figures 4.20 et 4.21.

Formellement, le bigraphe $S_{C_{mouvement}}$ est défini par :

$$S_{C_{mouvement}} : \epsilon \rightarrow \langle 2, \{x, z\} \rangle$$

tels que :

— $V^S_{C_{mouvement}} = \{Chambre, Personne, Couloir, Lampe_1, Lampe_2, Lampe_3, Moyen_1, Fort_2, Moyen_3\}$

— $E^S_{C_{mouvement}} = \{e_0, e1, e2, e3\}$

— $GP^S_{C_{mouvement}} : 0 \rightarrow 2$

— $GL^S_{C_{mouvement}} : \emptyset \rightarrow \{x, z\}$

— $I = \epsilon$ et $J = \langle 2, \{x, z\}\rangle$ où $n = 2$ et $Y = \{x, z\}$

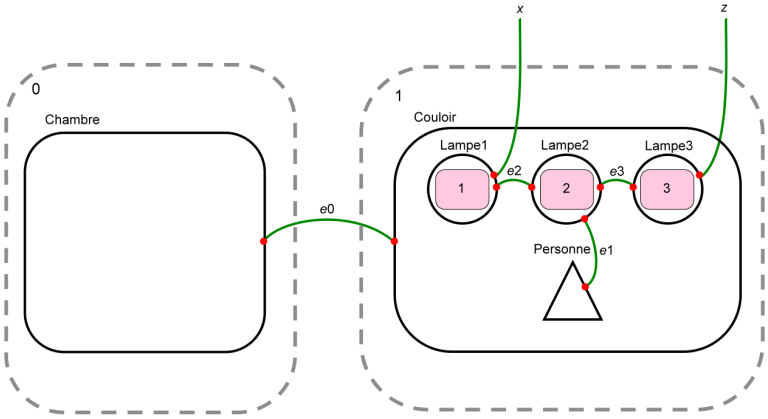


FIGURE 4.20 – Bigraphe S de l'état du mouvement

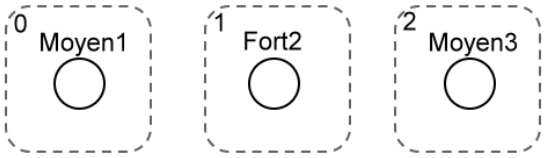


FIGURE 4.21 – Bigraphe $C_{mouvement}$ de l'état du mouvement

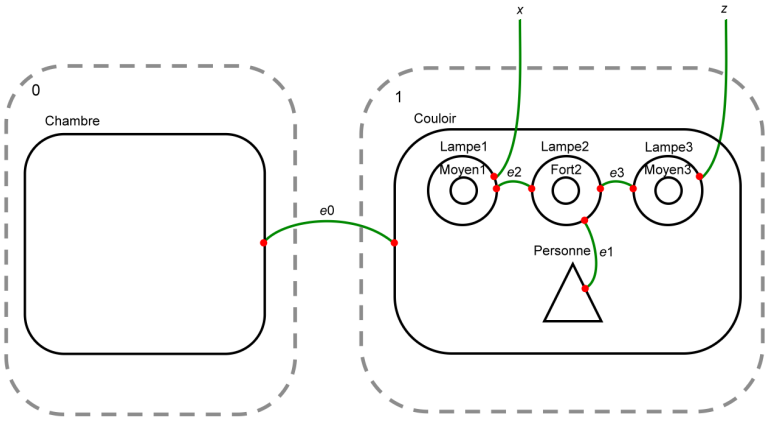


FIGURE 4.22 – Bigraphe de l'état du mouvement d'une personne

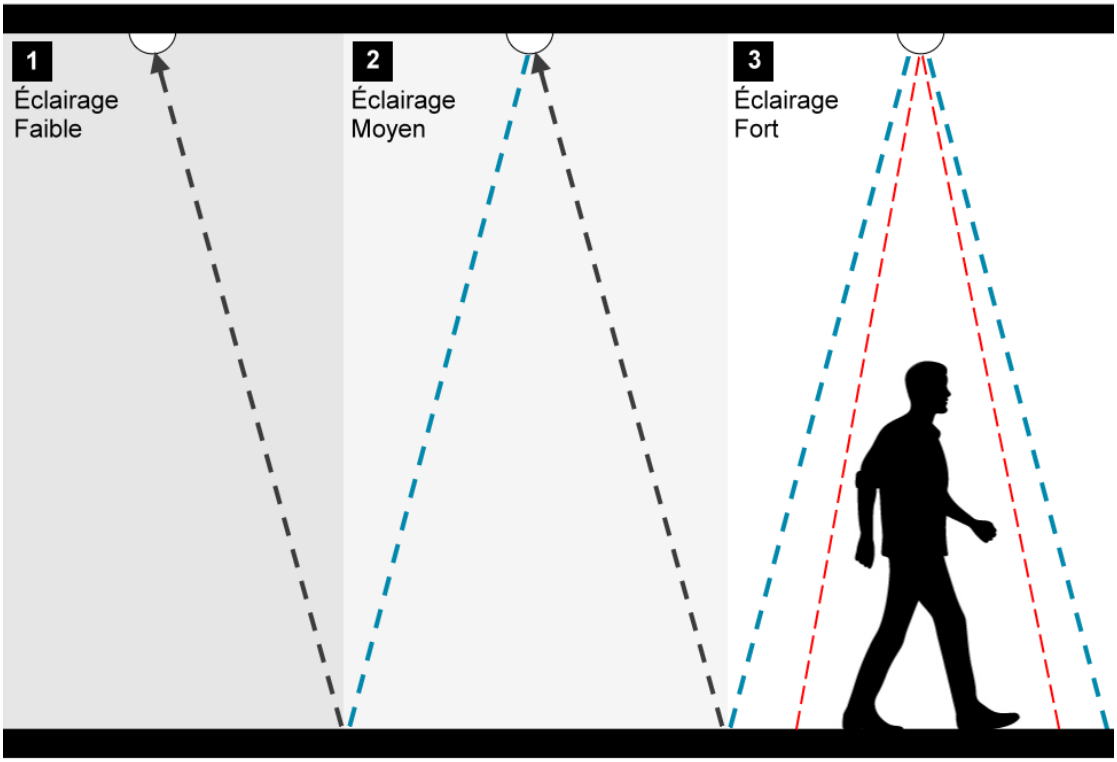


FIGURE 4.23 – Éclairage intelligent en fonction du mouvement continu

Événement (6) : Ajustement d'éclairage en fonction du mouvement continu.

$$Chambre \| Couloir.(Lampe_1.(Moyen_1) | Personne | Lampe_2.(Fort_2)) | z / Lampe_3.(Moyen_3)$$

$$\rightarrow$$

$$Chambre \| Couloir.(Lampe_1.(Faible_1) | Lampe_2.(Moyen_2)) | Personne | z / Lampe_3.(Fort_3)$$

De façon similaire, quand une personne entre dans la zone de la $lampe_3$ (figure 4.24), la $lampe_3$ passe automatiquement du mode $Moyen_3$ au mode $Fort_3$ et communique l'information du mouvement à sa lampe voisine $lampe_2$. Á la réception de l'information du mouvement, la $lampe_2$ et la $lampe_1$ basculent vers le mode $Moyen_2$ et $Faible_1$ respectivement.

La figure 4.26 représente le bigraphe $S_{C_{mouvement\_continu}}$ modélisant le système d'éclairage intelligent à l'état d'un mouvement continu d'une personne.

Le bigraphe $S_{C_{mouvement\_continu}}$ est le résultat de la composition des bigraphes $S \circ C_{mouvement\_continu}$ représentés respectivement par les figures 4.24 et 4.25, donné par :

$$S_{C_{mouvement\_continu}} : \epsilon \rightarrow \langle 2, \{x, z\} \rangle$$

où :

— $V^S_{C_{mouvement\_continu}} = \{Chambre, Personne, Couloir, Lampe_1, Lampe_2, Lampe_3, Faible_1, Moyen_2, Fort_3\}$

— $E^S_{C_{mouvement\_continu}} = \{e_0, e1, e2, e3\}$

— $GP^S_{C_{mouvement\_continu}} : 0 \rightarrow 2$

— $GL^S_{C_{mouvement\_continu}} : \emptyset \rightarrow \{x, z\}$

— $I = \epsilon$ et $J = \langle 2, \{x, z\}\rangle$ où $n = 2$ et $Y = \{x, z\}$

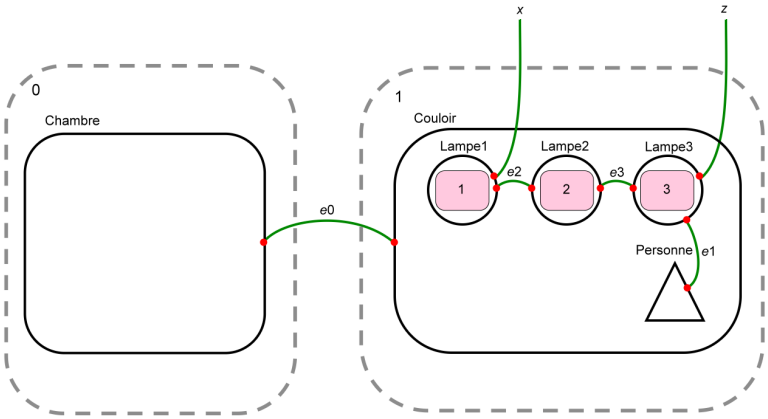


FIGURE 4.24 – Bigraphe S de l'état du mouvement continu

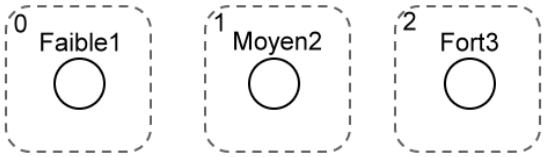


FIGURE 4.25 – Bigraphe $C_{mouvement\_continu}$ de l'état du mouvement continu d'une personne

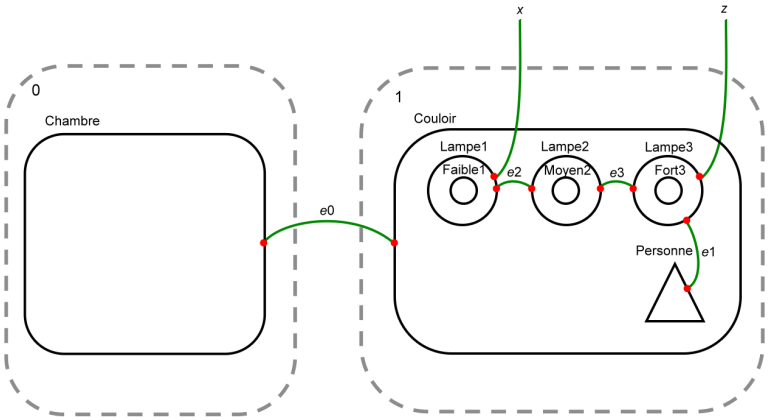


FIGURE 4.26 – Bigraphe de l'état du mouvement continu d'une personne

Événement (7) : Éteindre l'éclairage dès que la personne quitte le couloir.

$$Chambre \| Couloir.(Lampe_1.(Faible_1) | Lampe_2.(Moyen_2)) | Personne | z / Lampe_3.(Fort_3)$$

$$\rightarrow$$

$$Chambre \| Couloir.(Lampe_1.(\acute{E}teint_1) | Lampe_2.(\acute{E}teint_2)) | z / Lampe_3.(\acute{E}teint_3)$$

En cas d'absence, le système d'éclairage passe automatiquement à l'état initial (figure 4.17) en mettant toutes les lampes en mode Éteint$_i$.

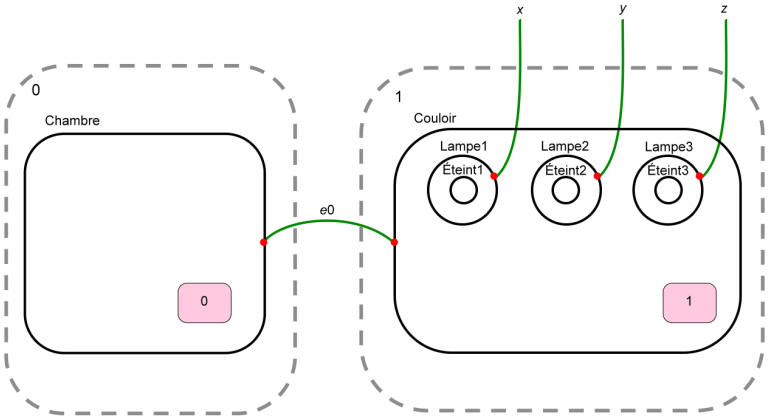


FIGURE 4.27 – Bigraphe de l'état de l'absence

Le bigraphe modélisant le système d'éclairage intelligent à l'état de l'absence d'une personne est donné par :

$$S_{C_{absence}} : \langle 2, \emptyset \rangle \rightarrow \langle 2, \{p, x, y, z\} \rangle$$

où :

— $V^S_{C_{absence}} = \{Chambre, Couloir, Lampe_1, Lampe_2, Lampe_3, \acute{E}teint_1, \acute{E}teint_2, \acute{E}teint_3\}$

— $E^S_{C_{absence}} = \{e_0\}$

— $GP^S_{C_{absence}} : 2 \rightarrow 2$

— $GL^S_{C_{absence}} : \emptyset \rightarrow \{p, x, y, z\}$

— $I = \langle 2, \emptyset \rangle$ où $m = 2$ et $X = \emptyset$

— $J = \langle 2, \{p, x, y, z\} \rangle$ où $n = 2$ et $Y = \{p, x, y, z\}$

Les sites 1 et 2 dans la figure 4.27 indiquent que le bigraphe peut héberger d'autres nœuds de type personne.

4.4 Avantages de BigCAS

Dans le chapitre 2, nous avons présenté une synthèse des approches liées à la modélisation des systèmes sensibles au contexte, et montré que les systèmes réactifs bigraphiques sont le formalisme le plus adapté à nos critères de modélisation. Cependant, les approches basées systèmes réactifs bigraphiques existantes présentent des limites dont les plus importantes sont : la complexité de la modélisation prédictive des systèmes multi-états [BDE+06, PKS12] et la redondance inutiles des informations de contexte [XXL11, WXL11]. De plus, ces approches prennent en compte seulement l'aspect structurel statique et pas l'aspect comportemental dynamique des systèmes sensibles au contexte.

En effet, notre modèle BigCAS a été conçu pour surmonter ces limites selon les principes suivants :

— Modèle dédié aux systèmes sensibles au contexte.
— Séparation complète entre la partie sensible au contexte et la partie non-sensible au contexte des systèmes.
— Modélisation à haut niveau d'abstraction des informations contextuelles.
— Prise en charge de l'aspect comportemental des systèmes sensibles au contexte.
— Extensibilité robuste pour répondre à l'aspect évolutif des systèmes sensibles au contexte.

4.5 Conclusion

Dans ce chapitre, nous avons présenté notre modèle BigCAS qui est un modèle à base de système réactifs bigraphiques dont le but est la modélisation à haut niveau d'abstraction de systèmes sensibles au contexte. Le modèle BigCAS est générique et adaptable a tout type de système sensible au contexte. Pour atteindre cet objectif, nous avons appliqué une technique de séparation entre la partie sensible au contexte et la partie non sensible au contexte. Tout d'abord, chaque partie a été modélisée séparément par un bigraphe distinct. Le bigraphe non-sensible au contexte S permet de modéliser la partie du système dont la structure est invariante, même si le contexte qui entoure le système varie. Alors que, le bigraphe sensible au contexte C_{id} modélise la partie du système affectée par les changements de contexte. Ensuite, nous avons appliqué l'opération de composition $S \circ C_{id}$ pour modéliser la structure générale du système.

En outre, l'aspect dynamique des systèmes sensibles au contexte est pris en charge par une nouvelle catégorie de règles de réaction bigraphiques. Ces dernières permettent de capturer les différentes reconfigurations internes et contextuelles intervenant dans le comportement des systèmes sensibles au contexte. Finalement, nous avons montré l'application de notre modèle à travers une étude de cas d'un système d'éclairage intelligent.

5 BigCAS-FA : Extension pour l'Analyse Formelle des Systèmes Sensibles au Contexte

Sommaire

5.1 Introduction

Aujourd'hui, les systèmes sensibles au contexte sont particulièrement utilisés dans tous les domaines critiques (médical, avionique, domotique, transport, etc.) pour lesquels une défaillance peut avoir des conséquences dramatiques en termes humain ou économique. La mise en œuvre de systèmes sûrs représente alors un véritable enjeu qui doit être pris en compte dès la phase de conception. Plusieurs approches existent pour valider le bon comportement de tels systèmes, telles que les tests classiques, les tests unitaires, la simulation, l'analyse de code, la vérification par preuves et la vérification formelle. La vérification formelle est l'approche robuste pour assurer qu'un système est correct. Il existe plusieurs techniques de vérification formelles, telles que la certification de code, les preuves de théorèmes et le model-checking.

En outre, le paradigme de contrat possède un lien très fort avec les méthodes formelles permettant de vérifier la correction du système et de ses composants vis-à-vis de leurs exigences. En effet, le contrat lui-même constitue une forme de spécification, mais, contrairement aux spécifications formelles, il n'est pas nécessaire de montrer explicitement que la spécification est bien réalisée par le système.

Dans ce chapitre, nous nous intéressons, dans un premier temps, à la vérification formelle par model-checking, qui est une approche automatique visant à déterminer si toutes les exécutions possibles d'un modèle respectent un comportement donné. Ensuite, nous étendons notre modèle BigCAS en incluant le paradigme de contrat comme un type d'analyse complémentaire dans lequel le système doit respecter une certaine invariance. Pour ce faire, nous proposons une extension appelée BigCAS-FA (Bigraphical Context-Aware System - Formal Analysis) permettant l'analyse formelle des systèmes sensibles au contexte. Tout d'abord, nous commençons par la vérification des certaines propriétés relatives à la sûreté et à la vivacité de tels systèmes en utilisant le model-checker BigMC [PDH12]. Ensuite, nous définissons des stratégies de reconfiguration à base de contrats qui consistent à guider le processus d'adaptation au contexte, puis, nous montrons comment BigCAS-FA intègre ces différentes stratégies.

5.2 Analyse des Propriétés Fonctionnelles

La technique de model-checking consiste à analyser un modèle du système pour vérifier qu'il respecte un ensemble de critères d'exigences définis généralement sous la forme de propriétés exprimées en logique temporelle. Les concepts de la logique temporelle sont assez faciles à utiliser, les opérateurs sont très proches en termes de langage naturel, ce qui rend la formalisation en logique temporelle assez simple. Bien que sous cette apparence simple, la formalisation en logique temporelle nécessite une expertise significative. Plus

précisément, le model-checking consiste à explorer tous les états possibles du système afin de vérifier s'ils répondent à la propriété désirée ou donner un contre-exemple indiquant de quelle façon le système peut violer cette propriété (figure 5.1). Cette technique est souvent accompagnée d'un outil puissant, appelé model-checker, qui peut être utilisé pour automatiser les diverses étapes de vérifications. Avec l'aide du model-checker, l'utilisateur peut trouver l'erreur et d'adapter le modèle ou la propriété pour empêcher la violation de cette dernière. Les propriétés exprimables en logique temporelle peuvent être classées en cinq grandes

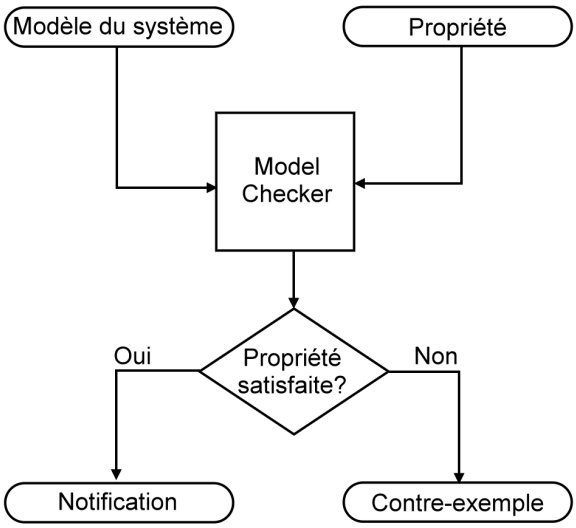


FIGURE 5.1 – Principe du model-checking

catégories : sûreté (*safety*), vivacité (*liveness*), atteignabilité (*reachability*), équité (*fairness*) et absence de blocage (*deadlock freeness*). En effet, toute propriété, exprimée en logique temporelle, en vertu du théorème de décomposition peut être réécrite sous la forme d'une conjonction de propriété de sûreté et de vivacité. Pour cela, dans cette section, nous nous focalisons sur la vérification de ces propriétés en utilisant le model-checker BigMC [PDH12] spécifiées sur des modèles de systèmes sensibles au contexte définis en BigCAS.

BigMC [PDH12](section 3.6.3) est un model-checker dédié aux bigraphes, semble être le seul outil qui permet de vérifier les propriétés d'un bigraphe tel que défini par ses fondateurs. Le principe de cet outil est de définir un modèle bigraphique en utilisant sa propre grammaire ainsi qu'une suite de règles de réaction associées à ce modèle. Pour vérifier une propriété, BigMC applique sur le modèle les règles de réaction définies jusqu'à ce qu'il rencontre, dans le meilleur cas, un état vérifiant la propriété ou il retourne un contre-exemple permettant ainsi d'identifier un cas d'exécution non conforme.

5.2.1 Sûreté

La propriété de sûreté a pour but de dire que « quelque chose de mauvais n'arrivera jamais ». Elle permet de définir les contraintes relatives aux différentes parties du système. La vérification d'une propriété de sûreté est simple et peut être réalisée par tous les model-checkers existants.

Bien qu'il n'existe pas de model-checker traitant la logique temporelle avec des opérateurs du passé, le formalisme de logique temporelle intégrant des opérateurs du passé et du futur est plus riche et donc facilite la transcription des énoncés en langue naturelle. Cependant, il est possible de vérifier les propriétés en logique temporelle contenant des opérateurs du passé en utilisant une des deux techniques suivantes :

— **Élimination du passé** : qui consiste à transformer la formule en logique temporelle de telle sorte que les opérateurs du passé disparaissent au profit d'opérateur du futur.

— **Utilisation des variables d'histoire (*History Variables*)** : qui consiste à transformer la propriété en logique temporelle en une négation de propriété d'atteignabilité en utilisant des variables d'histoires (variable de type booléen).

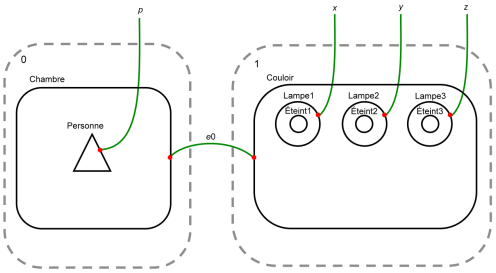


FIGURE 5.2 – Bigraphe d'état initial

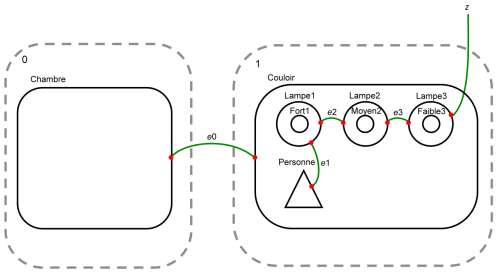


FIGURE 5.3 – Bigraphe d'état de présence

Le listing 5.1 montre la spécification du système d'éclairage intelligent exprimée en BigMC [PDH12].

Premièrement, le bigraphe d'état initial représenté dans la figure 5.2 est spécifié dans la partie modèle, alors que le bigraphe d'état de présence représenté dans la figure 5.3 est le résultat des quatre règles de réactions. Ensuite, pour assurer le correct fonctionnement du système d'éclairage intelligent à l'état de présence d'une personne dans le couloir, nous définissons une propriété de sûreté appelée « estAllumee », qui vérifie qu'à la présence d'une personne toutes les lampes doivent être allumées.

Listing 5.1– Vérification de propriété de sûreté

```
# Noeuds internes
%active Chambre : 1;
%active Personne : 1;
%active Couloir : 1;
%active Lampe : 2;

# Noeuds contextuels
%active Eteint : 0;
%active Faible : 0;
%active Moyen : 0;
%active Fort : 0;

# Noms externes
%name p;
%name x;
%name y;
%name z;

# Hyperarcs
%name e1;
%name e2;

# Règles de réaction (1) : Arrivée d'une personne.
Chambre[e0].Personne[p] || Couloir[e0].(Lampe[-,x].Eteint | Lampe[-,y].Eteint
   | Lampe[-,z].Eteint)
->
Chambre[e0]|| Couloir[e0].(Personne[e1] | Lampe[e1, x].$1 | Lampe[-,y].$2 |
  Lampe[-,z].$3);

# Règle de réaction (2): Détection de la présence d'une personne.
Personne[e1] || Lampe[e1, x].$1 -> Personne[e1] || Lampe[e1,x].Fort;

# Règle de réaction (3): Communication de l'information de la présence à les
  lampes voisines.
Lampe[e1,x].Fort || Lampe[-,y].$2 -> Lampe[e1,e2].Fort || Lampe[e2,y].Moyen;

# Règle de réaction (4): Ajustement d'éclairage en fonction de proximité.
Lampe[e2,y].Moyen || Lampe[-,z].$3 -> Lampe[e2,y].Moyen || Lampe[e2,z].Faible
  ;

# Modèle du système d'éclairage intelligent à l'état initial
Chambre[e0].Personne[p] | Couloir[e0].(Lampe[-,x].Eteint | Lampe[-,y].Eteint
  | Lampe[-,z].Eteint);

# Propriété
%property estAllumee !matches(Personne[e1]|Lampe[e1,x].Eteint);

%check;
```

Le listing ci-dessous montre le résultat de la vérification de la propriété de sûreté « estAllumee ». Nous constatons que le matching est atteint (Complete !) et la propriété est vérifiée après un nombre de transformations égal à 5.

Listing 5.2– Résultat de la vérification de la propriété estAllumee

```
1  Welcome to BigMC!
2  > /usr/local/bigmc/bin/bigmc -m 1000 -r 50 -p /var/folders/r5/
     _8wpv21n537g2qsqn0gnbsgw0000gn/T/bigmc_model4135411157822070342.bgm
3  1: (Chambre[e0].Personne[p].nil | Couloir[e0].(Lampe[-,x].Eteint.nil | Lampe
     [-,y].Eteint.nil | Lampe[-,z].Eteint.nil))
4  2: (Chambre[e0].nil | Couloir[e0].(Personne[e1].nil | Lampe[e1,x].nil | Lampe
     [-,y].nil | Lampe[-,z].nil))
5  3: (Chambre[e0].nil | Couloir[e0].(Personne[e1].nil | Lampe[e1,x].Fort.nil |
     Lampe[-,y].nil | Lampe[-,z].nil))
6  4: (Chambre[e0].nil | Couloir[e0].(Lampe[e1,e2].Fort.nil | Lampe[e2,y].Moyen.
     nil | Lampe[-,z].nil | Personne[e1].nil))
7  5: (Chambre[e0].nil | Couloir[e0].(Lampe[e1,e2].Fort.nil | Lampe[e2,y].Moyen.
     nil | Lampe[e2,z].Faible.nil | Personne[e1].nil))
8  [mc::step] Complete!
9  [mc::report] [q: 0 / g: 5] @ 6
```

5.2.2 Vivacité

La propriété de vivacité énonce que « quelque chose finira par avoir lieu ». Il s'agit d'une propriété plus forte que l'atteignabilité puisque l'atteignabilité indique qu'« il est possible que quelque chose se passe ».

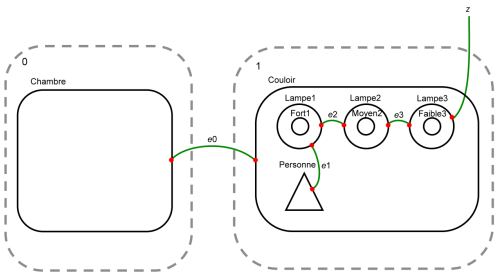


FIGURE 5.4 – Avant l'état de mouvement

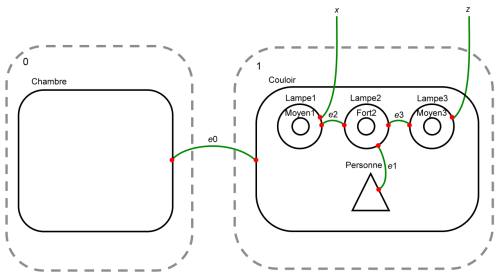


FIGURE 5.5 – Après l'état de mouvement

Dans la spécification ci-dessous (listing 5.3), le modèle initial représente le bigraphe de l'état de présence (figure 5.4), le résultat des trois règles de réaction exprime le bigraphe de l'état de mouvement représenté dans la figure 5.5 et la propriété de vivacité « movement » vérifie qu'il y a une lampe avec un éclairage fort pendant le passage de la personne dans le couloir.

Listing 5.3– Vérification de propriété de vivacité

```
# Noeuds internes
%active Chambre : 1;
%active Personne : 1;
%active Couloir : 1;
%active Lampe : 2;

# Noeuds contextuels
%active Eteint : 0;
%active Faible : 0;
%active Moyen : 0;
%active Fort : 0;

# Noms externes
%name p;
%name x;
%name y;
%name z;

# Règle de réaction (1): Mouvement d'une personne.
Chambre[e0]|| Couloir[e0].(Personne[e1] | Lampe[e1,e2].Fort | Lampe[e2,e3].
  Moyen | Lampe[e3,z].Faible);
->
Chambre[e0] || Couloir[e0].(Lampe[-,x].$1 | Personne[p] | Lampe[-,y].$2 |
  Lampe[-,z].$3)

# Règle de réaction (2): Détection de mouvement d'une personne.
Chambre[e0] || Couloir[e0].(Lampe[-,x].$1 | Personne[p] | Lampe[-,y].$2 |
  Lampe[-,z].$3)
->
Chambre[e0] || Couloir[e0].(Lampe[-,x].$1 | Personne[e1] | Lampe[e1,y].Fort |
   Lampe[-,z].$3)

# Règle de réaction (3): Communication de l'information de la présence à les
  lampes voisines.
Chambre[e0] || Couloir[e0].(Lampe[-,x].$1 | Personne[e1] | Lampe[e1,y].Fort |
   Lampe[-,z].$3)
->
Chambre[e0] || Couloir[e0].(Lampe[e2,x].Moyen | Personne[e1] | Lampe[e1,e2].
  Fort | Lampe[e2,z].Moyen)

# Modèle du système d'éclairage intelligent à l'état de présence
Chambre[e0]|| Couloir[e0].(Personne[e1] | Lampe[e1,e2].Fort | Lampe[e2,e3].
  Moyen | Lampe[e3,z].Faible);

# Propriété
%property mouvement matches(Lampe[e1,e2].Fort);
%check;
```

Le résultat de la vérification de la propriété de vivacité « mouvement » est représenté ci-dessous (listing 5.4). Nous remarquons que le processus de vérification est arrivé au bout (complete !) après 4 transformations subies par le modèle initial. Le fait que la vérification n'a pas abouti sur un contre-exemple, il y a toujours une lampe avec un éclairage fort quand la personne passe dans le couloir.

Listing 5.4– Résultat de la vérification de la propriété mouvement

```
1  Welcome to BigMC!
2  > /usr/local/bigmc/bin/bigmc -m 1000 -r 50 -p /var/folders/r5/\
     _8wpv21n537e2qsqg0gnbsgw0000gn/T/bigmc\_model2823873894351951450.bgm
3  1: (Chambre[e0].nil | Couloir[e0].(Personne[-].nil | Lampe[-,-].Fort.nil |
     Lampe[-,-].Moyen.nil | Lampe[-,z].Faible.nil))
4  2: (Chambre[e0].nil | Couloir[e0].(Personne[-].nil | Lampe[-,x].nil | Lampe
     [-,-].nil | Lampe[-,z].nil))
5  3: (Chambre[e0].nil | Couloir[e0].(Personne[-].nil | Lampe[-,x].nil | Lampe
     [-,-].Fort.nil | Lampe[-,z].nil))
6  4: (Chambre[e0].nil | Couloir[e0].(Personne[-].nil | Lampe[-,-].Moyen.nil |
     Lampe[-,y].Fort.nil | Lampe[-,z].Moyen.nil))
7  [mc::step] Complete!
8  [mc::report][q:0 / g: 4] @ 5
```

5.3 Reconfigurations Sensibles aux Contrats

Dans la section précédente, nous avons utilisé le model-checker BigMC [PDH12] pour étudier la faisabilité de notre modèle BigCAS. Ainsi, Il est également possible de vérifier d'autres propriétés de systèmes sensibles au contexte, telles que les propriétés de l'atteignabilité, l'équité et l'absence de blocage. Cependant, ce type d'analyse présente des limites en termes de prise en compte de la dynamicité et des aspects temporels et comportementaux de systèmes sensibles au contexte.

Pour surmonter ces limites, dans cette section, nous proposons une extension de BigCAS qui intègre des stratégies de reconfiguration à base de contrat. Mais avant d'introduire ces dernières, nous allons tout d'abord définir la notion de contrat.

5.3.1 Notion de Contrat

La conception par contrats (abrégée DbC pour Design by Contract en anglais) [Mey92a] est une approche de conception logicielle basée sur la théorie des types de données abstraits [Rua84] et sur le concept du contrat au sens juridique du terme. En avant, l'intérêt du DbC a été de spécifier précisément les interfaces des différents modules logiciels en termes de

pré-conditions, post-conditions et invariants.

En effet, les travaux sur les contrats remontent aux années 1960 avec Floyd [Flo67] et Hoare [Hoa69]. Ces travaux montrent l'intérêt de l'utilisation des assertions en programmation. Une assertion est une expression booléenne exprimant une propriété sémantique qui doit être vraie pendant l'exécution du programme. Par exemple, les assertions permettent le raffinement de contraintes de typage dans la spécification des interfaces de classe ou de composant.

Conceptuellement, les assertions ne sont qu'un mécanisme de base qui n'a que peu en commun avec les aspects méthodologiques et l'intégration au modèle objet qu'offrent les contrats. Cependant, plusieurs langages de programmation utilisent le mécanisme d'assertion pour mettre en œuvre les contrats avec des bibliothèques, des pré-processus ou d'autres moyens spécialisés.

Le paradigme de contrat a été bien accueillie par la communauté de génie logiciel. En effet, face à la diversité des entités logicielles et la complexité de leur fonctionnement, les contrats sont utilisés comme une partie de la spécification du système permettant d'introduire une certaine invariance que le système doit respecter. Cependant, le contrat n'est pas nécessairement complet dans le sens où il n'est pas possible de comprendre entièrement le système à partir des contrats parce qu'ils ne portent que sur une partie de la spécification du système.

Le principe central sur lequel repose le DbC est que les modules logiciels ont des responsabilités envers les autres modules avec lesquels ils interagissent. Ces responsabilités sont alors basées sur des règles formalisées entre ces modules. Des contrats sont créés pour chaque module dans le système avant la phase de codage. Les interactions entre les différents modules du système sont alors bornées par ces contrats.

Le travail le plus connu dans ce domaine est celui autour du langage Eiffel [Mey92b]. Eiffel est un langage de programmation orienté objet conçu par Bertrand Meyer. Il utilise des assertions dans des pré/post-conditions et dans des invariants de classes pour décrire des contrats entre l'utilisateur et une classe. D'autres langages plus récents comme le langage OCL (Object Constraint Language) [OCL06] pour UML ou JML (Java Modeling Language) [LRL+00] pour Java fournissent maintenant des mises en œuvre étendues de la conception par contrats.

Taxonomie des Contrats

Un contrat peut prendre diverses formes, selon les propriétés spécifiées et selon l'expression plus ou moins formelle de la spécification. Beugnard et al. [BJPW99] ont proposé une classification qui identifie quatre niveaux de contrat où chaque niveau contient les propriétés des contrats de niveau inférieur :

— **Syntaxique :** ce premier niveau s'applique à la forme syntaxique des opérations, généralement en définissant les noms des opérations que le composant peut effectuer, le type des paramètres d'entrées/sorties requises et les éventuelles exceptions qui pourraient survenir lors de l'exécution. Les langages de description d'interface (Interface Description Languages abrégés IDL) et les langages orientés objet typés peuvent servir à décrire ce niveau de contrat. En outre, il peut être statiquement vérifié.

— **Comportemental :** spécifie le comportement dynamique des opérations en utilisant des assertions booléennes (pré-conditions, post-conditions et invariants). Ces dernières sont liées à une opération, qui est vue comme une unité atomique. Ce niveau fixe la sémantique des opérations en précisant le typage des paramètres de chaque opération et les éventuelles dépendances entre leurs valeurs. Les travaux sur le DbC se situent essentiellement à ce niveau, avec des langages comme Eiffel [Mey92b], Sather [Omo91], OCL [OCL06] et iContract [Ens01].

— **Synchronisation :** il s'applique aux interactions dynamiques entre les composants en spécifiant des contraintes de synchronisation sur les appels de méthodes. Le but de ce type de contrat est la prise en compte des dépendances entre les services d'un composant tels que : la concurrence et le parallélisme.

— **Qualité de service :** toutes les propriétés non-fonctionnelles, c'est-à-dire celles qui n'apparaissent pas explicitement dans le système, telles que la performance, la sécurité, la disponibilité ou le temps de réponse correspondent à ce niveau de contrat. Le terme de Qualité de Service (Quality of Service abrégée QoS) est utilisé pour désigner ces propriétés. Cependant, les propriétés de qualité de service sont les plus complexes à exprimer et à analyser puisqu'elles dépendent de l'environnement et du contexte d'exécution qui ne sont pas spécifiés au moment de la description du système.

En outre, les niveaux de contrats sont de plus en plus négociables (figure 5.6) où un client peut plus difficilement demander un changement de type qu'un paramètre de qualité de service.

Cycle de Vie des Contrats

Les informations fournies par les composants lors de leur assemblage sont conservées dans un contrat de composition attaché aux interfaces de chaque composant. Ce contrat de composition est organisé selon les quatre niveaux présentés dans la section précédente (section 5.3.1). Lors de l'assemblage de composants, les contrats de composition sont comparés et forment un contrat s'ils sont compatibles entre eux. La vérification de compatibilité se fait à chaque niveau. Le contrat est violé si l'une des parties ne le satisfait plus. Cependant, celui-ci peut être renégocié selon le niveau du contrat falsifié. La figure 5.7 montre le cycle de vie d'un contrat. La renégociation d'un contrat peut être statique ou dynamique. La première correspond à une renégociation lors de la phase de spécification, alors qu'une renégociation

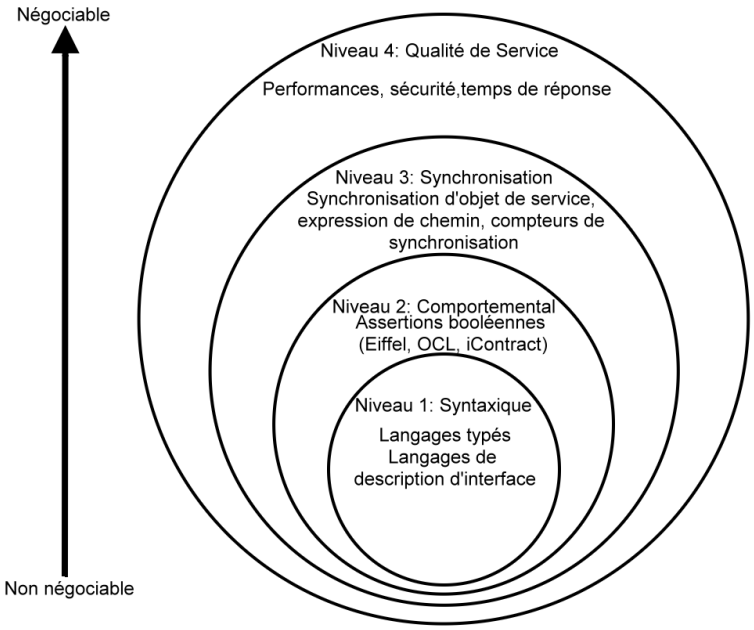


FIGURE 5.6 – Taxonomie des contrats

dynamique s'effectue à la phase d'exécution.

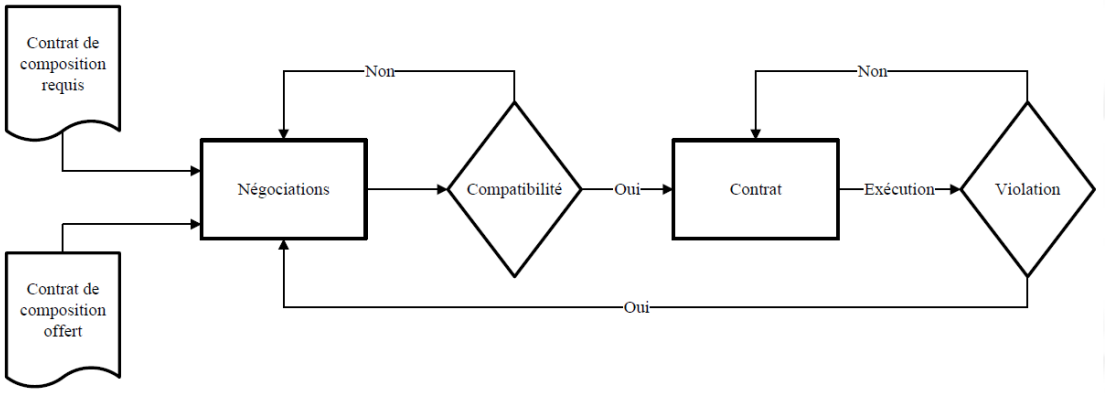


FIGURE 5.7 – Cycle de vie d'un contrat

5.3.2 Bigraphe du Contrat de Contexte

Un contrat de contexte est un contrat comportemental qui consiste en un ensemble de contraintes qui doivent être respectées par les éléments d'un système sensible au contexte. Chaque contrainte est une restriction associée à un élément particulier.

La définition formelle du bigraphe d'un contrat de contexte est donnée ci-dessous :

Définition 5.1 (Bigraphe T)**.** Étant donné un bigraphe $S_{C_{id}}$ modélisant un système sensible au contexte, dans un contexte id est défini par :

$$S_{C_{id}} = (V^S_{C_{id}}, E^S_{C_{id}}, ctrl^S_{C_{id}}, GP^S_{C_{id}}, GL^S_{C_{id}}) : I \to J$$

Un bigraphe T modélisant un contrat sur un système sensible au contexte est donné par :

$$T = (C_T, E_T, ctrl_T, G^P_T, G^L_T) : \epsilon \to J_T$$

tels que :

- C_T est un ensemble fini de nœuds où chaque nœud $c_i \in C_T$ est un nœud qui modélise une contrainte.
- E_T est un ensemble fini d'hyperarcs où chaque hyperarc $e_i \in E_T$ lie une contrainte $c_i \in C_T$ à un nœud $c_i \in V^S_{C_{id}}$.
- $ctrl_T = (\mathcal{K}_T, ar, \Phi) : C_T \to \mathcal{K}_T$ est une transformation qui associe à chaque nœud contrainte $c_i \in C_T$ un contrôle $k \in \mathcal{K}_T$ et une restriction $\varphi_i \in \Phi$ où

  $\forall c_i \in C_T, k_i : atomic \wedge ar(c_i) = 1$: un nœud contrainte est un nœud atomique contenant un seul port.
- φ_i est une restriction définie sur un nœud $v_i \in V^S_{C_{id}}$ qui doit être respectée en permanence (depuis la création jusqu'à la disparition du nœud).
- $G^P_T = (C_T, prnt_T, ctrl_T) : m_T \to n_T$ est le graphe de places associé à T, où

  $\forall c_i \in C_T, prnt_T(c_i) = \emptyset$: un nœud contrainte est un nœud sans parent hiérarchique.
- $G^L_T = (C_T, E_T, ctrl_T, link_T) : X_T \to Y_T$ est le graphe de liens de T, où $link_T : P_T \uplus X_T \to E_T \uplus Y_T$ est une transformation.
- ϵ représente l'origine du bigraphe T.
- $J_T = \langle n_T, Y_T \rangle$ représente l'interface externe du bigraphe T, où n_T est le nombre de régions et Y_T est l'ensemble des noms externes.

Exemple 5.1. Soit un bigraphe T (figure 5.8) modélisant un contrat sur un système sensible au contexte, donné par $T : \epsilon \to \langle 1, \{y_0\} \rangle$, tels que :

- $C_T = \{c_0\}$ représente l'ensemble des nœuds contraintes.
- $E_T = \emptyset$.
- $ctrl_T = \{(c_0 : 1, \varphi_0)\}$ indique la signature du nœud contrainte c_0.
- φ_0 est une restriction.
- $G^P_T : 0 \to 1$ et $G^L_T : \emptyset \to \{y_0\}$ sont les graphes de places et de liens de T respectivement.
- $I_T = \epsilon$ est l'origine du bigraphe T.
- $J_T = \langle 1, \{y_0\} \rangle$ est l'interface externe de T.

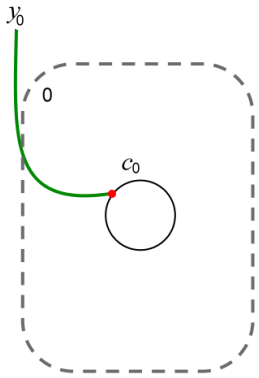


FIGURE 5.8 – Modélisation d'un bigraphe de contrat

5.3.3 Opérations sur les Contrats

Dans cette section, nous proposons des opérations sur l'ensemble des contraintes afin de décrire le comportement dynamique d'un contrat de contexte. Ces opérations sont illustrées par des exemples simples.

Ajout de Contraintes

L'opération d'ajout consiste à introduire de nouvelles contraintes sur les éléments du système sensible au contexte.

Définition 5.2 (ADD_{c_i})**.** L'ajout d'une contrainte est modélisé par une règle de réaction $ADD_{c_i} = (R : m_T \rightarrow J_T, R' : m'_T \rightarrow J_T)$ où $R : m_T \rightarrow J_T$ est un bigraphe redex et $R' : m'_T \rightarrow J_T$ est un bigraphe reactum, tel que :

$$ADD(c_i) = C_T \cup \{c_i\}$$

Exemple 5.2. La figure 5.9 représente une règle de réaction ADD appliquée au bigraphe T (figure 5.8) dont la forme est :

$$ADD_{c_1} = (R : 0 \rightarrow \{y_0\}, R' : 0 \rightarrow \{y_0, y_1\})$$

L'application de la règle ADD_{c_1} a comme effet l'ajout du nœud contrainte c_1.

L'expression algébrique correspondante à la règle de réaction ADD_{c_1} est comme suit :

$$ADD_{c_1} : y_0 / c_0 \rightarrow y_0 y_1 / c_0 \mid c_1$$

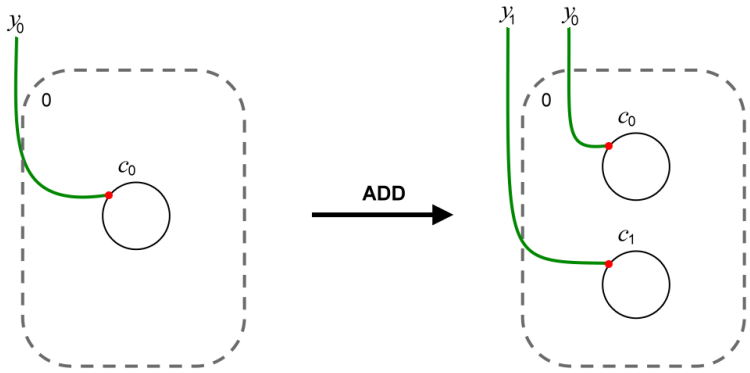


FIGURE 5.9 – Ajout d'une contrainte

Suppression de Contraintes

L'opération de suppression consiste à éliminer les contraintes inutiles sur les éléments du système sensible au contexte.

Définition 5.3 ($DROP_{c_i}$)**.** La suppression d'une contrainte est modélisée par une règle de réaction $DROP_{c_i} = (R : m_T \rightarrow J_T, R' : m'_T \rightarrow J_T)$ où $R : m_T \rightarrow J_T$ est un bigraphe redex et $R' : m'_T \rightarrow J_T$ est un bigraphe reactum, tel que :

$$DROP(c_i) = C_T \setminus \{c_i\}$$

Exemple 5.3. Soit $DROP_{c_1}$ (figure 5.10) une règle de réaction appliquée au bigraphe T représenté dans la figure 5.8 et donnée par :

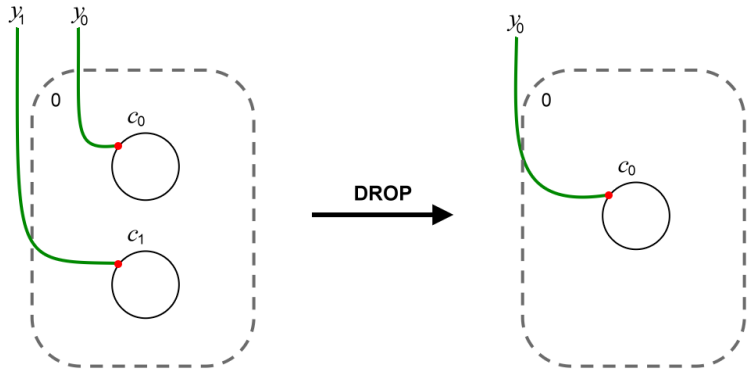


FIGURE 5.10 – Suppression d'une contrainte

$$DROP_{c_1} = (R : 0 \rightarrow \{y_0, y_1\}, R' : 0 \rightarrow \{y_0\})$$

La règle de réaction $DROP_{c_1}$ consiste en la suppression du nœud contrainte c_1 et toutes ses connexions.

L'expression algébrique de cette règle de réaction est donnée ci-dessous :

$$DROP_{c_1} : y_0 y_1 / c_0 \mid c_1 \rightarrow y_0 / c_0$$

Modification de Contraintes

L'opération de modification consiste à mettre à jour les spécifications des restrictions déjà établies.

Définition 5.4 ($UPDATE_{c_i}$)**.** La modification d'une contrainte est modélisée par une règle de réaction $UPDATE_{c_i} = (R : m_T \rightarrow J_T, R' : m'_T \rightarrow J_T)$ où $R : m_T \rightarrow J_T$ est un bigraphe redex et $R' : m'_T \rightarrow J_T$ est un bigraphe reactum, tel que :

$$UPDATE(c_i) = C_T \setminus \{c_i\} \cup \{c_i^*\} / ctrl_T(c_i) = (k_i, \varphi_i) \wedge ctrl_T(c_i^*) = (k_i, \varphi_i^*)$$

Exemple 5.4. Soit $UPDATE_{c_0}$ (figure 5.11) une règle de réaction appliquée au bigraphe T et donnée comme suit :

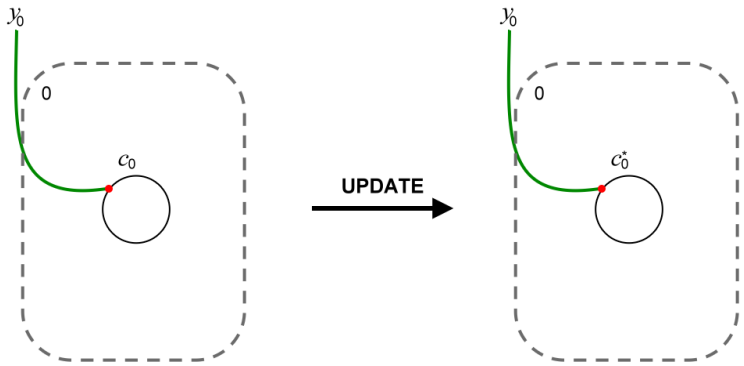


FIGURE 5.11 – Modification d'une contrainte

$$UPDATE_{c_0} = (R : 0 \rightarrow \{y_0\}, R' : 0 \rightarrow \{y_0\})$$

La règle de réaction $UPDATE_{c_0}$ consiste en la modification de la restriction φ_0 du nœud contrainte c_0.

L'expression algébrique de la règle de réaction $UPDATE_{c_0}$ est donnée par :

$$UPDATE_{c_0} : y_0 / c_0 \rightarrow y_0 / c_0^*$$

5.3.4 Contractualisation des Systèmes Sensibles au Contexte

L'objet de cette section est de décrire l'extension BigCAS-FA sur lequel se base le travail présenté dans ce chapitre. Nous commençons par introduire la définition formelle de la contractualisation d'un système sensible au contexte. Ensuite nous illustrons ce concept à travers un exemple.

Définition 5.5 (Bigraphe $S^T_{C_{id}}$)**.** Étant donné un bigraphe $S_{C_{id}}$ modélisant un système sensible au contexte, dans un contexte id est défini par :

$$S_{C_{id}} = (V^S_{C_{id}}, E^S_{C_{id}}, ctrl^S_{C_{id}}, GP^S_{C_{id}}, GL^S_{C_{id}}) : I \to J$$

Un bigraphe $S^T_{C_{id}}$ modélisant un système sensible au contexte avec un contrat T est défini par :

$$S^T_{C_{id}} \stackrel{\text{def}}{=} (S \parallel T) \circ C_{id}$$

où

$$S^T_{C_{id}} = (V^T_{S_{C_{id}}}, E^T_{S_{C_{id}}}, ctrl^T_{S_{C_{id}}}, GP^T_{S_{C_{id}}}, GL^T_{S_{C_{id}}}) : I \to J$$

— $V^T_{S_{C_{id}}} = C_T \uplus V^S_{C_{id}}$ est un ensemble fini de nœuds dans un contexte id donné par l'union disjointe de l'ensemble des nœuds contraintes C_T, l'ensemble des nœuds système V_S et l'ensemble des nœuds contextuels $V_{C_{id}}$.

— $E^T_{S_{C_{id}}} = E_T \uplus E^S_{C_{id}}$ est un ensemble fini d'hyperarcs dans un contexte id donné par l'union disjointe de l'ensemble des hyperarcs E_T, l'ensemble des hyperarcs système E_S et l'ensemble des hyperarcs contextuels $E_{C_{id}}$.

— $\mathcal{K}^T_{S_{C_{id}}} = \mathcal{K}_T \uplus \mathcal{K}^S_{C_{id}}$ est une signature étendue, définie par un ensemble des contrôles où $ctrl^T_{S_{C_{id}}} : V^T_{S_{C_{id}}} \to \mathcal{K}^T_{S_{C_{id}}}$ est une nouvelle transformation qui associe à chaque nœud $v_i \in V^T_{S_{C_{id}}}$ un contrôle $k \in \mathcal{K}^T_{S_{C_{id}}}$.

— $GP^T_{S_{C_{id}}} = G^P_T \parallel GP^S_{C_{id}}$ est le graphe de places associé à $S^T_{C_{id}}$ donné par le produit parallèle des graphes de places $G^P_T : m_T \to n_T$ et $GP^S_{C_{id}} : m_S \to n_S$, où sa fonction de parenté $prnt^T_{S_{C_{id}}} = prnt_T \uplus prnt^S_{C_{id}}$ est définie par : Si $w \in k \uplus V^T_{S_{C_{id}}}$ est un site ou un nœud dans $S^T_{C_{id}}$ alors

$$prnt^T_{S_{C_{id}}}(w) \stackrel{\text{def}}{=} \begin{cases} prnt^S_{C_{id}}(w) & \text{si } w \in k \uplus V_{S_{C_{id}}} \text{ et } prnt^S_{C_{id}}(w) \in V_{S_{C_{id}}}, \\ prnt_T(j) & \text{si } w \in k \uplus V_{S_{C_{id}}} \text{ et } prnt^S_{C_{id}}(w) = j \in m_T, \\ prnt_T(w) & \text{si } w \in C_T. \end{cases}$$

— $GL^T_{S_{C_{id}}} = G^L_T \parallel GL^S_{C_{id}}$ est le graphe de liens de $S^T_{C_{id}}$, donné par le produit parallèle des graphes de liens $G^L_T : X_T \to Y_T$ et $GL^S_{C_{id}} : X^S_{C_{id}} \to Y^S_{C_{id}}$, où $link^T_{S_{C_{id}}} = link_T \uplus link^S_{C_{id}}$

est une transformation définie par : Si $q \in X \uplus P_T \uplus P^S_{C_{id}}$ est un point de $S_{C_{id}} \otimes T$ alors

$$link(q) \stackrel{\text{def}}{=} \begin{cases} link^S_{C_{id}}(q) & \text{si } q \in X \uplus P^S_{C_{id}} \text{ et } ink^S_{C_{id}}(q) \in E_{S_{C_{id}}}, \\ link_T(y) & \text{si } q \in X \uplus P^S_{C_{id}} \text{ et } link^S_{C_{id}}(w) = y \in Y_T, \\ link_T(q) & \text{si } q \in P_T. \end{cases}$$

.

— $I = \langle m, X \rangle$ et $J = \langle n, Y \rangle$ représentent respectivement les interfaces internes et externes du bigraphe $S^T_{C_{id}}$, où $m = m^S_{C_{id}}$ est le nombre de sites, $X = X^S_{C_{id}}$ est l'ensemble des noms internes, $n = n_T + n^S_{C_{id}}$ est le nombre de régions, et $Y = Y_T \bigcup Y^S_{C_{id}}$ est l'ensemble des noms externes.

Exemple 5.5. Soit un bigraphe d'un système sensible au contexte $S_{C_{exemple}} : \epsilon \to \langle 2, \{y_0\} \rangle$ (figure 4.7) et un bigraphe du contrat $T : \epsilon \to \langle 1, \{y_0\} \rangle$ (figure 5.8). Le produit parallèle des bigraphes $S_{C_{exemple}}$ et T est un bigraphe $S^T_{C_{exemple}} \stackrel{\text{def}}{=} S_{C_{exemple}} \parallel T$ (figure 5.12) modélisant la structure du système sensible au contexte sous contrat, donné par :

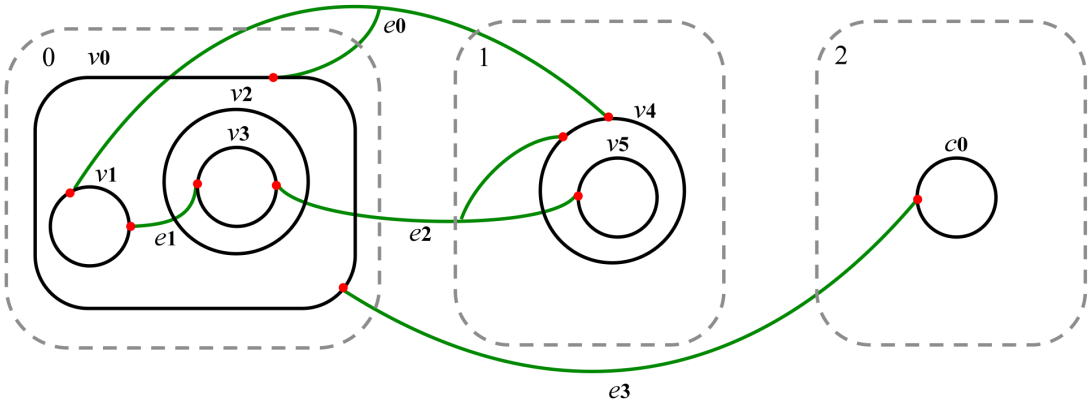


FIGURE 5.12 – Bigraphe $S^T_{C_{exemple}} : \epsilon \to \langle 3, \emptyset \rangle$ sensible au contrat

$$S^T_{C_{exemple}} : \epsilon \to \langle 3, \emptyset \rangle$$

tels que :

— $V^T_{S_{C_{exemple}}} = \{v_0, v_1, v_2, v_3, v_4, v_5, c_0\}$

— $E^T_{S_{C_{exemple}}} = \{e_0, e_1, e_2, e_3\}$

— $ctrl^T_{S_{C_{exemple}}} = \{v_0 : 2 v_1 : 3, v_2 : 1, v_3 : 2, v_4 : 2, v_5 : 1, c_0 : 1\}$

— $prnt^T_{S_{C_{exemple}}} = \{v_0 : \emptyset, v_1 : v_0, v_2 : v_0, v_3 : v_2, v_4 : \emptyset, v_5 : v_4, c_0 : \emptyset\}$

— $I = \epsilon$

— $J = \langle 3, \emptyset \rangle$

Le graphe de places représenté dans la figure 5.13 est le résultat du produit parallèle des graphes de places $GP^{S}_{C_{exemple}} \parallel G^{P}_{T} = 0 \rightarrow 3$ et le graphe de liens $GL^{T}_{S_{C_{exemple}}} : \emptyset \rightarrow \emptyset$ (figure 5.14) est le résultat du produit parallèle des graphes de liens $GL^{S}_{C_{exemple}} \parallel G^{L}_{T}$.

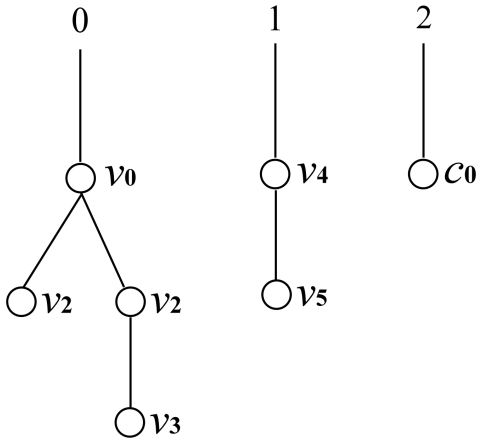


FIGURE 5.13 – Graphe de places $GP^{T}_{S_{C_{exemple}}} : 0 \rightarrow 3$

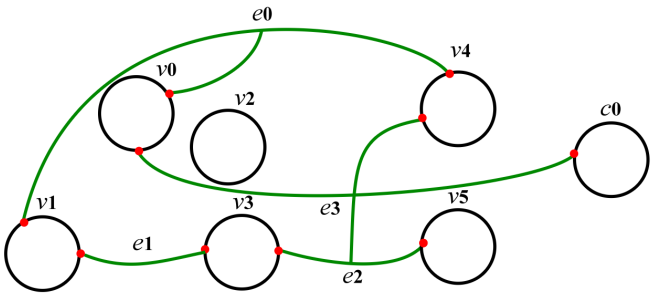


FIGURE 5.14 – Graphe de liens $GL^{T}_{S_{C_{exemple}}} : \emptyset \rightarrow \emptyset$

5.3.5 Avantages de la Conception par Contrat

Les principaux avantages que présente la conception par contrat dans la modélisation d'un système sensible au contexte sont les suivants :

— **Réutilisation :** notre approche vise à favoriser la réutilisation de modèles de systèmes sensibles au contexte. Dans ce sens, il est très intéressant d'introduire les contrats au niveau du contrôle des nœuds.

— **Séparation des préoccupations :** le système sensible au contexte et son contrat sont représentés par deux bigraphes distincts.

— **Possibilité d'outillage :** grâce à une formalisation précise des langages de contrats, il est possible d'outiller convenablement la technique afin de générer du code pour garantir certaines propriétés des contrats.

— **Documentation :** les invariants et les pré/post-conditions d'un contrat peuvent être utilisés comme des éléments de base de la documentation lors de sa génération automatique à partir du code et de ses commentaires, comme le permettent des outils disponibles dans de nombreux langages.

5.4 Étude de Cas : Contraintes de l'Éclairage Intelligent

Dans cette section, nous illustrons l'extension BigCAS-FA en introduisons des contraintes sur le fonctionnement du système d'éclairage intelligent. Premièrement, nous commençons par introduire, à l'état initial du système, un nouveau port pour tout nœud sensible au contrat. Ensuite, nous présentons trois scénarios périodiques liés au fonctionnement du système d'éclairage intelligent.

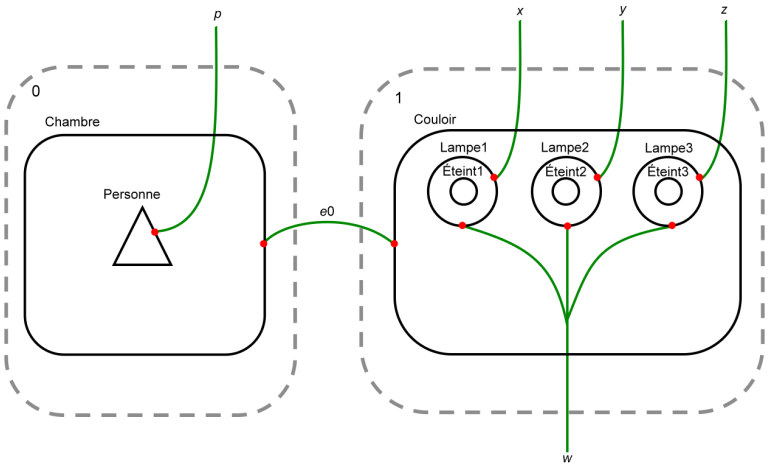


FIGURE 5.15 – Bigraphe de l'état initial du système d'éclairage intelligent

Tout d'abord, le bigraphe formalisant le système d'éclairage intelligent à l'état initial avant l'arrivée d'une personne (figure 5.15) est donné par :

$$S_{C_{initial}} : \langle 0, \{w\} \rightarrow \langle 2, \{p, x, y, z\} \rangle$$

où :

— $V^S_{C_{initial}} = \{Chambre, Personne, Couloir, Lampe_1, Lampe_2, Lampe_3, \text{É}teint_1, \text{É}teint_2, \text{É}teint_3\}$

— $E^S_{C_{initial}} = \{e_0\}$

— $GP^S_{C_{initial}} : 0 \rightarrow 2$ et $GL^S_{C_{initial}} : \{w\} \rightarrow \{p,x,y,z\}$

— $I = \langle 0, \{w\}\rangle$ où $m = 0$ et $X = \{w\}$

— $J = \langle 2, \{p,x,y,z\}\rangle$ où $n = 2$ et $Y = \{p,x,y,z\}$

Dans la figure 5.15, nous remarquons que les nœuds $\mathsf{lampe}_1, \mathsf{lampe}_2$ et lampe_3 ont un nouveau port lié au nom interne w. Ces nœuds sont appelés des nœuds sensibles au contrat.

5.4.1 Éclairage de Jour

La première contrainte consiste à assurer pendant la journée un éclairage naturel suffisant qui puisse remplacer totalement l'éclairage artificiel.

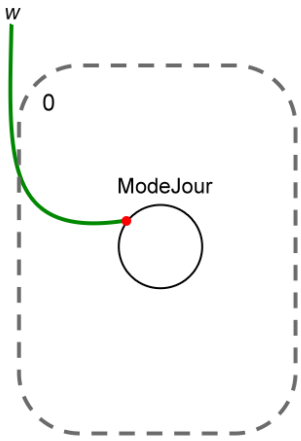


FIGURE 5.16 – Bigraphe de contrat de jour

Le bigraphe contrat $Jour$ présenté dans la figure ci-dessus est donné par :

$$Jour : \epsilon \rightarrow \langle 1, \{w\}\rangle$$

où :

— $C_{Jour} = \{ModeJour\}$

— $E_{Jour} = \emptyset$.

— $ctrl_{Jour} = \{(ModeJour : 1, \varphi_{ModeJour})\}$

— $\varphi_{ModeJour} : \{Heure > 7:00AM \quad \text{et} \quad Heure < 8:00PM\}$

— $G^P_{Jour} : 0 \rightarrow 1$ et $G^L_{Jour} : \emptyset \rightarrow \{w\}$ sont les graphes de places et de liens de $Jour$ respectivement.

— $I_{Jour} = \epsilon$ et $J_{Jour} = \langle 1, \{w\}\rangle$

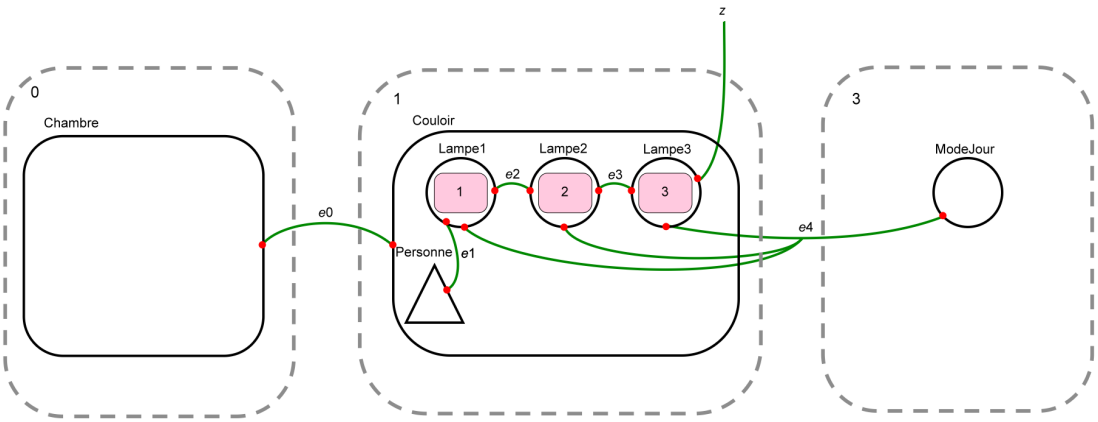


FIGURE 5.17 – Bigraphe de l'état de initial pendant la journée

La figure 5.17 représente le bigraphe formalisant l'état initial du système d'éclairage intelligent avant l'arrivée d'une personne.

Le bigraphe formalisant le système d'éclairage intelligent à l'état initial pendant la journée est donné par :

$$S^{Jour}_{C_{initial}} : \langle 3, \emptyset \rangle \rightarrow \langle 3, \{z\} \rangle$$

où :

— $V^{Jour}_{S_{C_{initial}}} = \{Chambre, Personne, Couloir, Lampe_1, Lampe_2, Lampe_3, ModeJour\}$

— $E^{Jour}_{S_{C_{initial}}} = \{e_0, e_1, e2, e_3, e_4\}$

— $ctrl^{Jour}_{S_{C_{initial}}} = \{Chambre : 1, Personne : 1, Couloir : 1, Lampe_1 : 4, Lampe_2 : 3, Lampe_3 : 3, ModeJour : 1\}$

— $GP^{Jour}_{S_{C_{initial}}} : 3 \rightarrow 3$ et $GL^{Jour}_{S_{C_{initial}}} : \emptyset \rightarrow \{z\}$

— $I = \langle 3, \emptyset \rangle$ où $m = 3$ et $X = \emptyset$

— $J = \langle 3, \{z\} \rangle$ où $n = 3$ et $Y = \{z\}$

La règle de réaction décrivant l'état de présence d'une personne pendant la journée est donnée par :

$$Chambre.(p/Personne) \| Couloir.(x/Lampe_1.(\acute{E}teint_1) | y/Lampe_2.(\acute{E}teint_2) | z/Lampe_3.(\acute{E}teint_3)) \| ModeJour$$

$$\rightarrow$$

$$Chambre \| Couloir.(Personne | Lampe_1.(\acute{E}teint_1) | Lampe_2.(\acute{E}teint_2) | z/Lampe_3.(\acute{E}teint_3)) \| ModeJour$$

Le bigraphe représenté dans la figure 5.18 modélise la partie sensible au contexte à l'état de la présence d'une personne pendant la journée et donné par :

$$C_{présence} : \epsilon \rightarrow \langle 3, \emptyset \rangle$$

où :

— $V^{C_{présence}} = \{\acute{E}teint_1, \acute{E}teint_2, \acute{E}teint_3\}$

— $E^{C_{présence}} = \emptyset$

— $G^P_{C_{présence}} : 0 \rightarrow 3$ et $G^L_{C_{présence}} : \emptyset \rightarrow \emptyset$

— $I = \epsilon$ et $K = \langle 3, \emptyset \rangle$ où $k = 3$ et $Z = \emptyset$

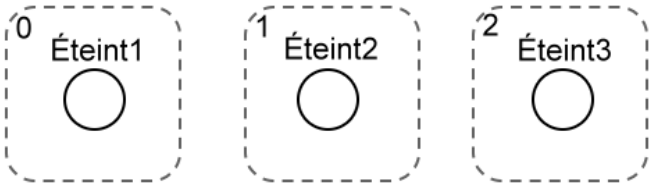


FIGURE 5.18 – Bigraphe $C_{présence}$ en mode de jour

La figure 5.19 représente le bigraphe $S^{Jour}_{C_{présence}}$ modélisant la structure générale du système d'éclairage intelligent à l'état de la présence d'une personne.

Formellement, le bigraphe $S^{Jour}_{C_{présence}}$ est le résultat du produit parallèle $S_{C_{présence}} \parallel Jour$ tel que :

$$S^{Jour}_{C_{présence}} : \epsilon \rightarrow \langle 3, \{z\} \rangle$$

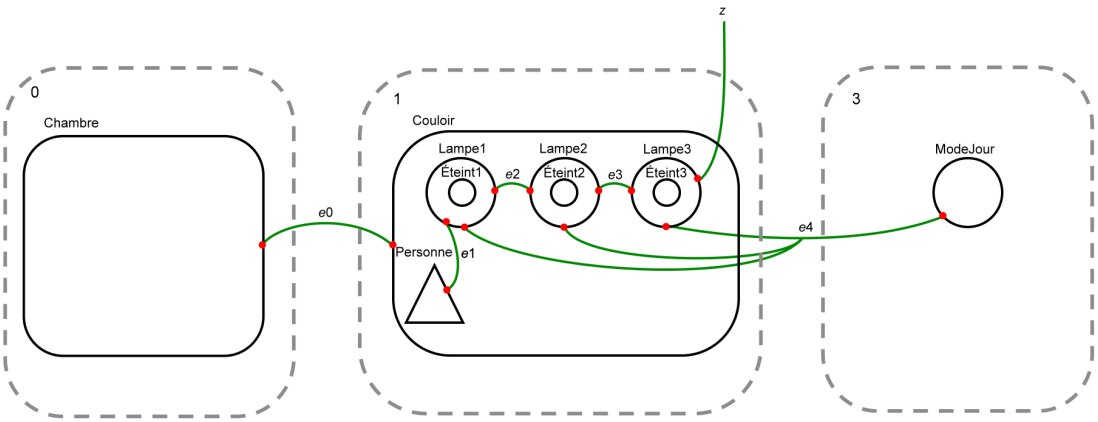


FIGURE 5.19 – Bigraphe de l'état de la présence d'une personne pendant la journée

Nous remarquons que le système d'éclairage intelligent reste à l'état initial même avec la présence d'une personne pendant les heures de la journée.

5.4.2 Éclairage de Nuit

La contrainte de nuit consiste à appliquer le scénario d'éclairage en fonction de la présence décrit dans le chapitre précédent (section 4.3.1) pendant les heures de nuit.

$$UPDATE_{Mode} = (R : 0 \rightarrow \{w\}, R' : 0 \rightarrow \{w\})$$

La règle de réaction $UPDATE_{Mode}$ consiste en la modification de la restriction $\varphi_{ModeJour}$ telle que : $\varphi_{ModeJour} \rightarrow \varphi_{ModeNuit}$ où : $\varphi_{ModeNuit} : \{Heure > 8:00PM \quad \text{et} \quad Heure < 7:00AM\}$.

L'expression algébrique de la règle de réaction $UPDATE_{Mode}$ (Figure 5.20) est donnée par :

$$UPDATE_{Mode} : w/ModeJour \rightarrow w/ModeNuit$$

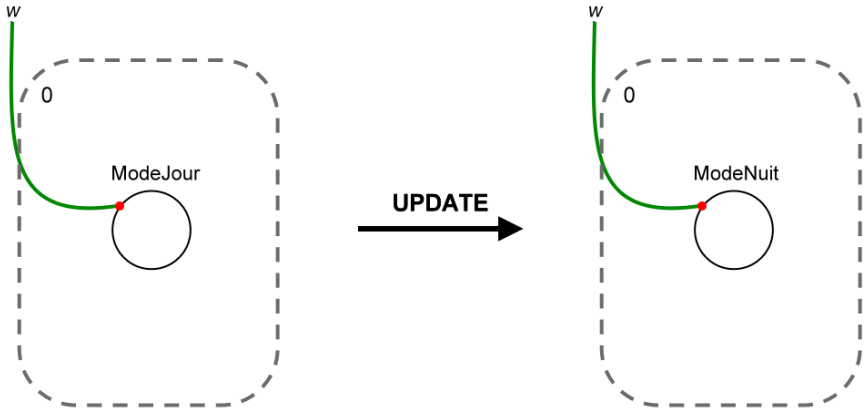


FIGURE 5.20 – Règle de réaction de changement de mode d'éclairage

Le bigraphe contrat $Nuit$ présenté dans la partie droite de la règle de réaction (figure 5.20) est donné par :

$$Nuit : \epsilon \rightarrow \langle 1, \{w\} \rangle$$

— $C_{Nuit} = \{ModeNuit\}$

— $E_{Nuit} = \emptyset$.

— $ctrl_{Nuit} = \{(ModeNuit : 1, \varphi_{ModeNuit})\}$

— $\varphi_{ModeNuit} : \{Heure > 8:00PM \quad \text{et} \quad Heure < 7:00AM\}$

— $G^P_{Nuit} : 0 \rightarrow 1$ et $G^L_{Nuit} : \emptyset \rightarrow \{w\}$

— $I_{Nuit} = \epsilon$ et $J_{Nuit} = \langle 1, \{w\} \rangle$

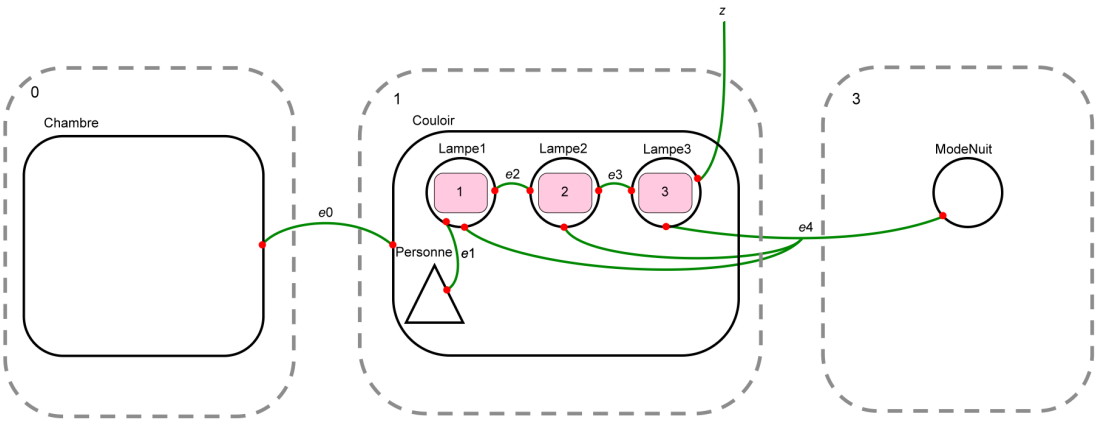


FIGURE 5.21 – Bigraphe de l'état de initial pendant la nuit

La règle de réaction décrivant l'état de présence d'une personne pendant la nuit est donnée par :

$$Chambre.(p/Personne) \| Couloir.(x/Lampe_1.(\acute{E}teint_1) | y/Lampe_2.(\acute{E}teint_2) | z/Lampe_3.(\acute{E}teint_3)) \| ModeNuit$$

$$\rightarrow$$

$$Chambre \| Couloir.(Personne | Lampe_1.(Fort_1) | Lampe_2.(Moyen_2) | z/Lampe_3.(Faible_3)) \| ModeNuit$$

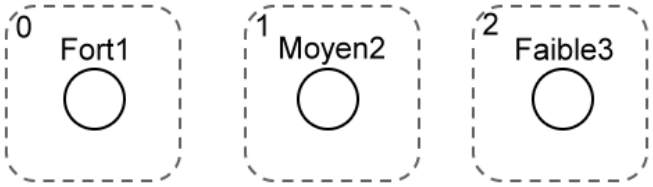


FIGURE 5.22 – Bigraphe $C_{présence}$ en mode de nuit

La figure 5.22 modélise la partie sensible au contexte à l'état de la présence d'une personne $C_{présence}$ pendant la nuit. Formellement, le bigraphe $C_{présence}$ est donné par :

$$C_{présence} : \epsilon \rightarrow \langle 3, \emptyset \rangle$$

où :

— $V^{C_{présence}} = \{Fort_1, Moyen_2, Faible_3\}$

— $E^{C_{présence}} = \emptyset$

— $G^P_{C_{présence}} : 0 \rightarrow 3$ et $G^L_{C_{présence}} : \emptyset \rightarrow \emptyset$

— $I = \epsilon$ et $K = \langle 3, \emptyset \rangle$ où $k = 3 \quad$ et $\quad Z = \emptyset$

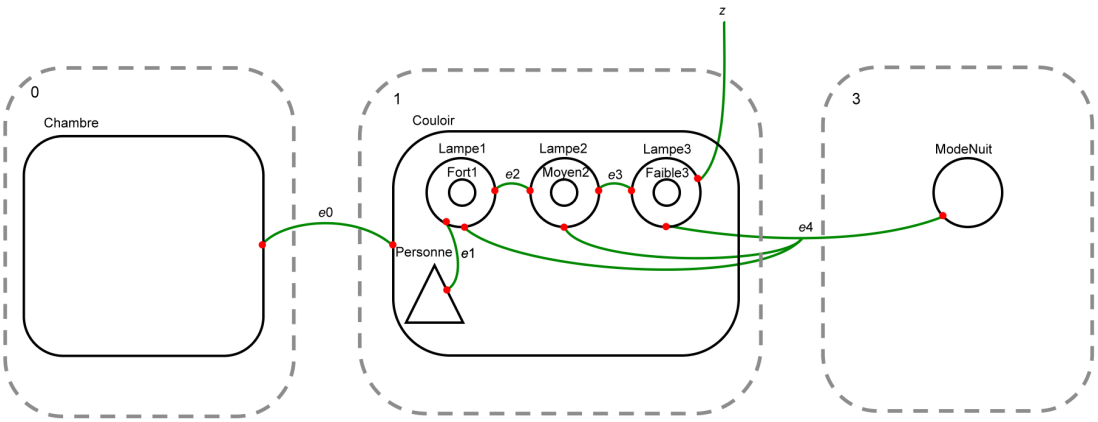


FIGURE 5.23 – Bigraphe de l'état de la présence d'une personne pendant la nuit

Le bigraphe représenté dans la figure 5.23 modélise le système d'éclairage intelligent à l'état de présence d'une personne pendant la nuit et donné par :

$$S^{Nuit}_{C_{présence}} : \epsilon \rightarrow \langle 3, \{z\} \rangle$$

où :

— $V^{Nuit}_{S_{C_{présence}}} = \{Chambre, Personne, Couloir, Lampe_1, Lampe_2, Lampe_3, Fort_1, Moyen_2, Faible_3, ModeNuit\}$

— $E^{Nuit}_{S_{C_{présence}}} = \{e_0, e_2, e_3, e_4\}$

— $ctrl^{Nuit}_{S_{C_{présence}}} = \{Chambre : 1, Personne : 1, Couloir : 1, Lampe_1 : 3, Lampe_2 : 3, Lampe_3 : 3, Fort_1 : 0, Moyen_2 : 0, Faible_3 : 0, ModeNuit : 1\}$

— $GP^{Nuit}_{S_{C_{présence}}} : 0 \rightarrow 3$ et $GL^{Nuit}_{S_{C_{présence}}} : \emptyset \rightarrow \{z\}$

— $I = \epsilon$ et $J = \langle 3, \{z\} \rangle$ où $n = 3$ et $Y = \{z\}$

5.4.3 Eco-éclairage

Afin de favoriser l'économie d'énergie, la contrainte eco-éclairage consiste à éteindre la lumière en cas d'absence de détection de présence de plus de dix minutes pendant la nuit.

$$ADD_{TempsAbsence} = (R : 0 \rightarrow \{w\}, R' : 0 \rightarrow \{w\})$$

La règle de réaction $ADD_{TempsAbsence}$ (figure 5.24) consiste à ajouter une contrainte de temps pour éteindre la lumière en cas d'absence d'une personne de plus de dix minutes.

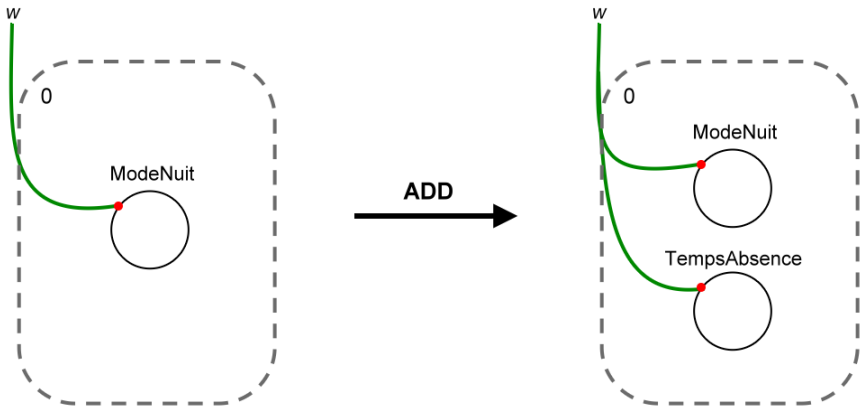


FIGURE 5.24 – Règle de réaction d'ajustement du temps d'absence

Formellement, la restriction $\varphi_{TempsAbsence}$ est donnée par :

$$\varphi_{TempsAbsence} : \{Temps >= 10 minutes\}$$

L'expression algébrique de la règle de réaction $ADD_{TempsAbsence}$ (figure 5.24) est donnée par :

$$ADD_{TempsAbsence} : w/ModeNuit \rightarrow w/ModeNuit \quad | \quad TempsAbsence$$

Le bigraphe contrat Eco représenté dans la figure 5.24 est donné par :

$$Eco : \epsilon \rightarrow \langle 1, \{w\} \rangle$$

— $C_{Eco} = \{ModeNuit, TempsAbsence\}$

— $E_{Eco} = \emptyset$.

— $ctrl_{Eco} = \{(ModeNuit : 1, \varphi_{ModeNuit}), (TempsAbsence : 1, \varphi_{TempsAbsence})\}$

— $\varphi_{ModeNuit} : \{Heure > 8 : 00PM \quad \text{et} \quad Heure < 7 : 00AM\}$

— $\varphi_{TempsAbsence} : \{Temps >= 10 minutes\}$

— $G^P_{Eco} : 0 \rightarrow 1$ et $G^L_{Eco} : \emptyset \rightarrow \{w\}$

— $I_{Eco} = \epsilon$ et $J_{Eco} = \langle 1, \{w\} \rangle$

La règle de réaction décrivant l'état d'absence d'une personne de plus de 10 minutes

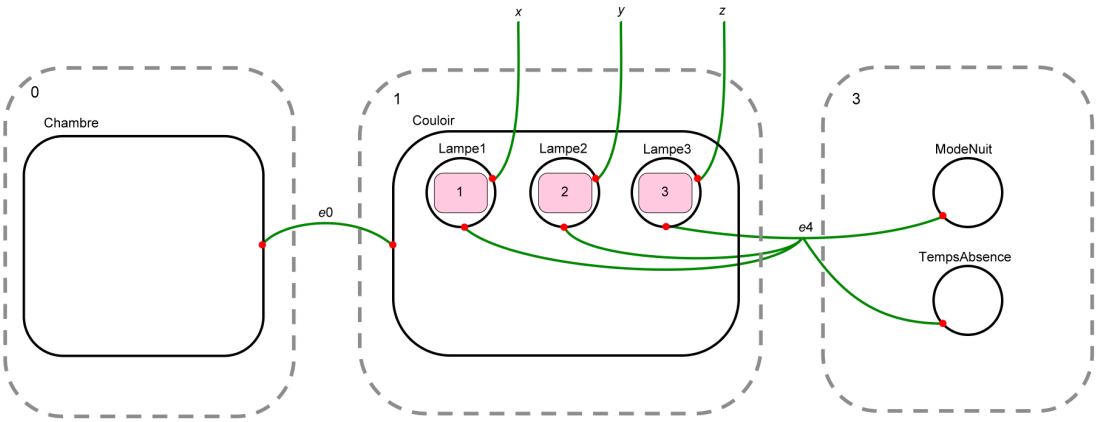


FIGURE 5.25 – Bigraphe initial d'éco-eclairage

pendant la nuit est donnée par :

$$Chambre \| Couloir.(Lampe_1.(Faible_1) | Lampe_2.(Moyen_2)) | Personne | z / Lampe_3.(Fort_3) \| ModeNuit | TempsAbsence$$

$$\rightarrow$$

$$Chambre \| Couloir.(Lampe_1.(\acute{E}teint_1) | Lampe_2.(\acute{E}teint_2)) | z / Lampe_3.(\acute{E}teint_3) \| ModeNuit | TempsAbsence$$

Les figures 5.26 et 5.27 représentent les bigraphes redex et reactum de $C_{absence}$ respectivement.

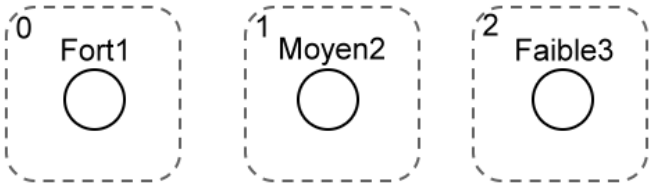


FIGURE 5.26 – Bigraphe $C_{absence}$ en mode de nuit de moins de 10 minutes

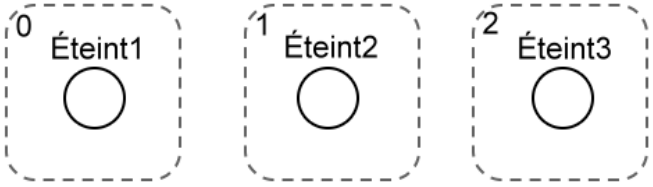


FIGURE 5.27 – Bigraphe $C_{absence}$ en mode de nuit après 10 minutes

Le bigraphe de la figure 5.28 modélise le système d'éclairage intelligent à l'état d'absence

d'une personne pendant la nuit et donné par :

$$S^{Eco}_{C_{absence}} : \epsilon \rightarrow \langle 3, \{x, y, z\}\rangle$$

où :

— $V^{Eco}_{S_{C_{absence}}} = \{Chambre, Couloir, Lampe_1, Lampe_2, Lampe_3, \acute{E}teint_1, \acute{E}teint_2, \acute{E}teint_3, ModeNuit, TempsAbsence\}$

— $E^{Eco}_{S_{C_{absence}}} = \{e_0, e_4\}$

— $ctrl^{Eco}_{S_{C_{absence}}} = \{Chambre : 1, Couloir : 1, Lampe_1 : 2, Lampe_2 : 2, Lampe_3 : 2, \acute{E}teint_1 : 0, \acute{E}teint_2 : 0, \acute{E}teint_3 : 0, ModeNuit : 1, TempsAbsence : 1\}$

— $GP^{Eco}_{S_{C_{absence}}} : 0 \rightarrow 3$ et $GL^{Eco}_{S_{C_{absence}}} : \emptyset \rightarrow \{x, y, z\}$

— $I = \epsilon$ et $J = \langle 3, \{z\}\rangle$ où $n = 3$ et $Y = \{x, y, z\}$

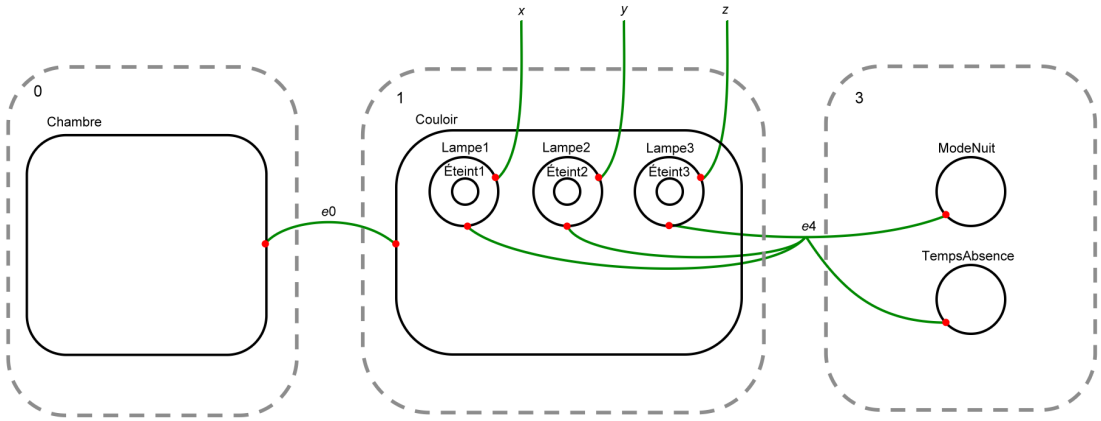


FIGURE 5.28 – Bigraphe d'éco-éclairage

5.5 Conclusion

Dans ce chapitre, nous avons proposé une extension à BigCAS, appelée BigCAS-FA (Bigraphical Context-Aware System - Formal Analysis), permettant la vérification formelle des systèmes sensibles au contexte. Tout d'abord, nous avons utilisé le model-checker BigMC (Bigraphical Model Checker) [PDH12]) pour vérifier certaines propriétés relatives à la sûreté et à la vivacité spécifiées sur un modèle de système sensible au contexte défini en BigCAS. Ensuite, nous avons introduit des stratégies de reconfiguration à base de contrats qui consistent principalement en un ensemble de règles de réactions bigraphiques déclenchées par des événements conduisant à une adaptation sensible au contexte. Les stratégies de reconfiguration proposées ont été prise en charge par BigCAS-FA.

6 BigCAS-Tool : Plateforme pour la Modélisation des Systèmes Sensibles au Contexte

Sommaire

6.1 Introduction

Ce chapitre présente BigCAS-Tool (Bigraphical Context-Aware System - Tool), une plateforme permettant de spécifier, modéliser et vérifier les systèmes sensibles au contexte exprimés sous forme de bigraphe. Tout d'abord, la section 6.2 introduit les besoins qui ont motivé la création de la plateforme BigCAS-Tool. Ensuite, la section 6.3 présente l'architecture globale de cette plateforme. Dans la section 6.4, nous présentons le développement de la plateforme BigCAS-Tool. Puis, dans la section 6.5, nous montrons comment rendre cette plateforme accessible à d'autres outils, ainsi que comment l'étendre pour introduire de nouvelles fonctionnalités spécifiques à des besoins particuliers dans la section 6.6. Enfin, dans la section 6.7, nous illustrons la plateforme BigCAS-Tool et ses fonctionnalités à travers l'étude de cas du système d'éclairage intelligent présentée dans le chapitre 4.

6.2 Motivation

Le chapitre 2 a souligné les manques actuels de la prise en compte des spécificités des systèmes sensibles au contexte par les approches existantes. Parallèlement, nous avons mis en évidence que les systèmes réactifs bigraphiques sont le formalisme le mieux adapté pour combler ces lacunes. Á ce titre, nous avons proposé deux modèles : BigCAS et BigCAS-FA prenant en compte les caractéristiques nécessaires pour la modélisation et la vérification des systèmes sensibles au contexte respectivement. Afin de valider pratiquement nos propositions, il est nécessaire de mettre en œuvre une plateforme à base de ces modèles bigraphiques pour la conceptualisation et l'analyse des systèmes sensibles au contexte.

En effet, grâce au projet BPL (*Bigraphical Programming Language Project*) [1], quelques outils ont été déjà élaborés afin de permettre la modélisation des systèmes réactifs bigraphiques. Cependant, la majorité de ces outils, tels que BPL Tool [HG11, GDBH11], BigMC [PDH12] et BigraphER [Sev12] s'appuient sur le langage des termes pour la représentation des bigraphes ignorant éventuellement l'aspect graphique de ce formalise, alors que cela est particulièrement destiné aux experts du domaine et nécessite un apprentissage théorique pour une personne novice.

Bigraphspace [GLM] est l'une des premières tentatives visant à représenter graphiquement les systèmes réactifs bigraphiques afin de simplifier la manipulation de tel formalisme. Néanmoins, ce projet n'a pas été totalement finalisé, et aucun outil n'existe pour l'instant.

En 2013, Faithfull et al. [FPH13] ont développé Big Red, le premier prototype d'éditeur graphique dont le but est de rendre les systèmes réactifs bigraphiques plus accessibles et

1. www.itu.dk/research/theory/bpl/

conviviaux même aux utilisateurs débutants. Pour cela, les points suivants ont été abordés :

— **Interfaçage** : Big Red peut s'interfacer facilement avec les outils de bigraphe existants.

— **Accessibilité** : les modèles de bigraphe décrits sous Big Red peuvent être exportés au format XML, ce qui permet d'être utilisés aisément par n'importe quel langage et n'importe quelle plateforme, sans subir de dépendances.

— **Extensibilité** : Big Red est un outil open source qui permet l'intégration des spécificités propres aux domaines d'application des systèmes réactifs bigraphiques.

Par conséquent, notre choix s'est directement orienté vers Big Red, qui correspond le plus à nos besoins techniques, que nous avons étendu pour développer une plateforme dédiée à la spécification et la vérification des systèmes sensibles au contexte.

6.3 Architecture de BigCAS-Tool

BigCAS-Tool est une extension de l'outil Big Red [FPH13] permettant la manipulation des modèles bigraphiques spécifiques aux systèmes sensibles au contexte. Big Red est un éditeur développé sous Eclipse qui permet la modélisation des systèmes réactifs bigraphiques d'une manière visuelle. Il étend la plateforme Eclipse avec les formats de fichiers qui représentent : les objets d'un système réactif bigraphique, les assistants de création des fichiers de modèles, et les éditeurs pour modifier ces modèles.

En effet, Big Red définit plusieurs points d'extension, ceux-ci permettent de l'étendre et d'ajouter des supports pour les nouveaux outils externes. La figure 6.1 montre le graphe de dépendances de BigCAS-Tool où les cercles et les rectangles représentent des plugins et des groupes de plugins respectivement. Les figures de contours en pointillés font partie de la plateforme Eclipse, tandis que celles avec des contours solides sont spécifiques à Big Red et à BigCAS-Tool. Les flèches représentent les dépendances entre les différents plugins. Généralement, la plupart des dépendances de BigCAS-Tool sont simplement des composants de base de la plateforme Eclipse (figure 6.2) utilisés pour concevoir l'interface utilisateur :

— Eclipse Platform Runtime (OSGi : Open Services Gateway initiative) est le noyau d'exécution qui gère l'ensemble des plugins.

— Les éditeurs graphiques sont développés à base du framework GMF (Graphical Modeling Framework).

— Les autres contributions de l'interface utilisateur, comme les boîtes à outils, sont développées en utilisant les plugins JFace et SWT (Standard Widget Toolkit).

— Eclipse Workbench est la couche permettant d'organiser et manipuler les différents composants graphiques, tels que les vues, les éditeurs et la perspective de BigCAS-Tool.

Dans la section suivante, nous présentons notre contribution qui consiste à étendre l'outil Big Red en se basant principalement sur les points suivants :

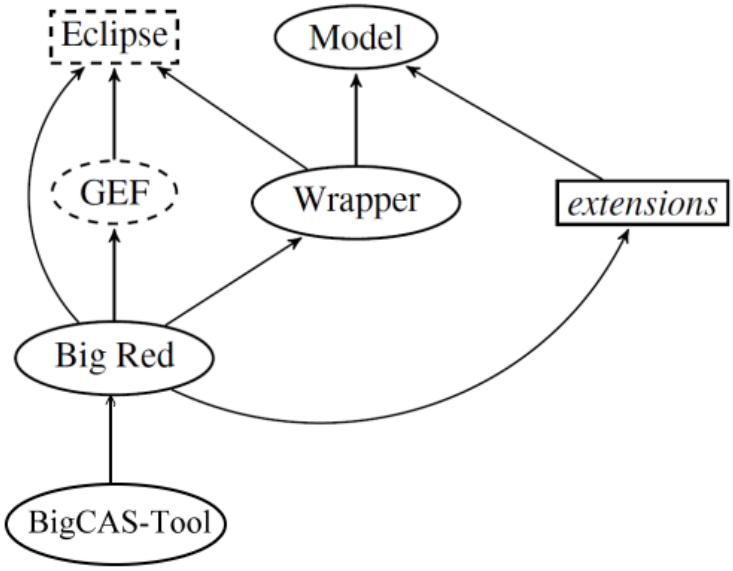


FIGURE 6.1 – Graphe de dépendance de BigCAS-Tool

— Développement d'une interface utilisateur pour modéliser séparément la partie sensible au contexte et la partie non-sensible contexte d'un système ubiquitaire.

— Introduction d'un mécanisme de typage de nœuds et de liens (c'est-à-dire *internes* et *contextuels*).

— Séparation entre les règles de réaction internes et contextuelles.

— Introduction de l'opération de composition pour la génération automatique du bigraphe qui modélise l'ensemble du système sensible au contexte.

6.4 Développement de BigCAS-Tool

Le modèle BigCAS est supporté par un éditeur graphique intégré à Eclipse sous forme d'un ensemble de plugins. Le choix d'Eclipse est dicté par le fait que c'est un environnement open source et largement utilisé pour construire des plateformes de développement ouvertes et extensibles.

Dans ce qui suit, nous présentons les outils qui nous ont servi pour le développement de notre plateforme BigCAS-Tool.

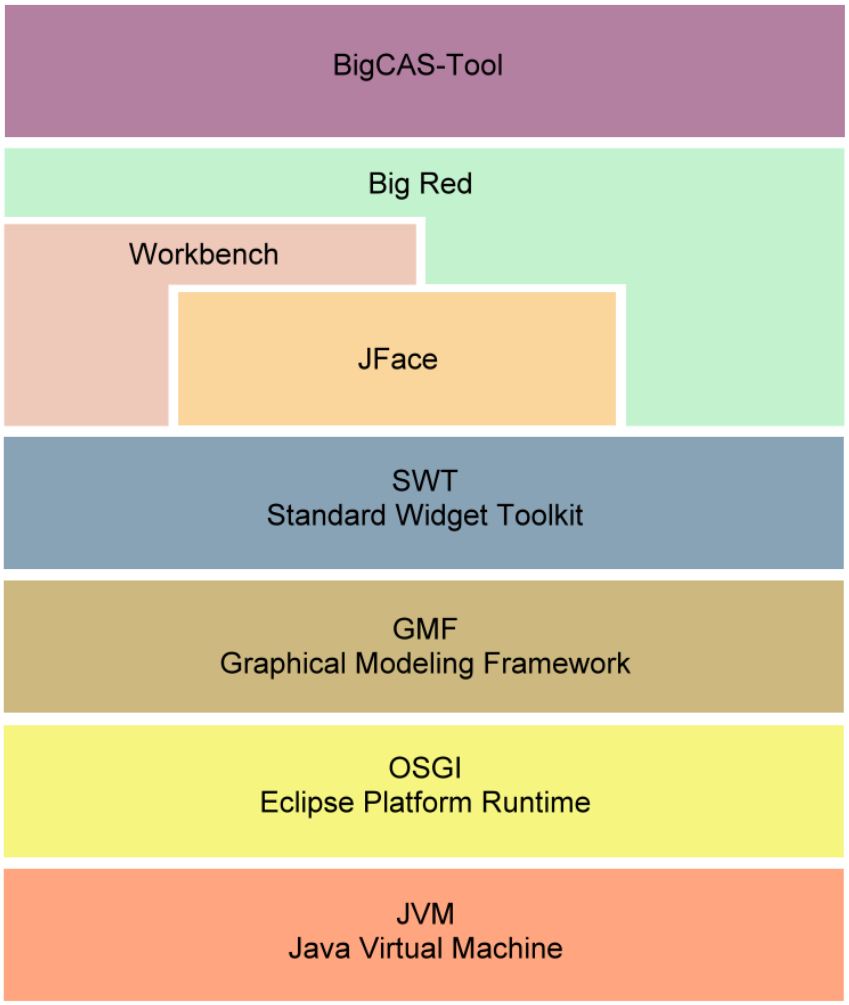


FIGURE 6.2 – Architecture de BigCAS-Tool

Eclipse

Eclipse [2] est un environnement de développement intégré (Integrated Development Environment ou IDE), compatible avec la presque totalité des langages de programmation actuels. Il offre aux développeurs un espace graphique et convivial de travail. La spécificité d'Eclipse vient du fait que son architecture est totalement développée autour de briques logicielles extensibles (communément appelées plugins) dont le but est de fournir un environnement modulaire permettant de réaliser aisément le développement de différents types d'applications.

Les plugins Eclipse utilisés pour développer la plateforme BigCAS-Tool sont les suivants :

2. http://www.eclipse.org/

GMF (Graphical Modeling Framework)

GMF [3] est un projet dédié à la communauté Eclipse supportant l'évolution et la promotion des technologies de développement dirigées par les modèles. GMF est basé sur les frameworks EMF (Eclipse Modeling Framework) [4] et GEF (Graphical Editing Framework) [5] qui fournissent une infrastructure permettant de simplifier le développement des prototypes des éditeurs graphiques grâce à la génération automatique du code source Java. Le framework EMF permet la modélisation et la génération automatique ou semi-automatique du code Java à partir des méta-modèles Ecore spécifiés en utilisant différentes syntaxes : Java annoté, schéma XML, diagramme de classe UML, etc. Le méta-modèle Ecore, utilisé par EMF, est compatible avec le standard MOF (Meta Object Facility) proposé par l'OMG (Object Management Group) [6]. Alors que le framework GEF prend en compte à la fois la modélisation des concepts d'un domaine et leur notation en fournissant le support graphique requis pour générer un éditeur riche à partir d'un méta-modèle EMF.

SWT (The Standard Widget Toolkit)

SWT [7] est une bibliothèque graphique libre qui encapsule des composants graphiques (panel, label, texte, bouton, etc.), des widgets pour développer les interfaces graphiques, et une implémentation native spécifique à chaque système d'exploitation qui sera utilisée à l'exécution de l'application. Cependant, SWT est une API de bas niveau proposant des objets qui permettent la création d'interfaces graphiques mais qui nécessitent énormément de code.

JFACE

JFACE [8] est une bibliothèque graphique qui s'appuie sur SWT pour faciliter son utilisation dans le développement des applications autonomes (*standalone*) en fournissant des éléments de plus haut niveau. Elle encapsule un certain nombre de traitements et réduit ainsi la quantité de code à produire. JFACE est une boîte à outils dans laquelle on retrouve notamment : des vues fournissant les abstractions des widgets SWT (listes, tables, arborescences, etc.), la définition de comportements au travers des actions, les assistants de préférences (boîtes de dialogues) et les support de ressources (images, polices, couleurs, etc.).

La figure 6.3 représente l'interface graphique de BigCAS-Tool sous la plateforme Eclipse. Un projet BigCAS-Tool, comme le montre la figure 6.4, contient quatre dossiers principaux :

3. http://www.eclipse.org/swt/
4. https://www.eclipse.org/modeling/emf/
5. https://eclipse.org/gef/
6. http://www.omg.org
7. http://www.eclipse.org/swt/
8. http://eclipse.org/modeling/gmp/

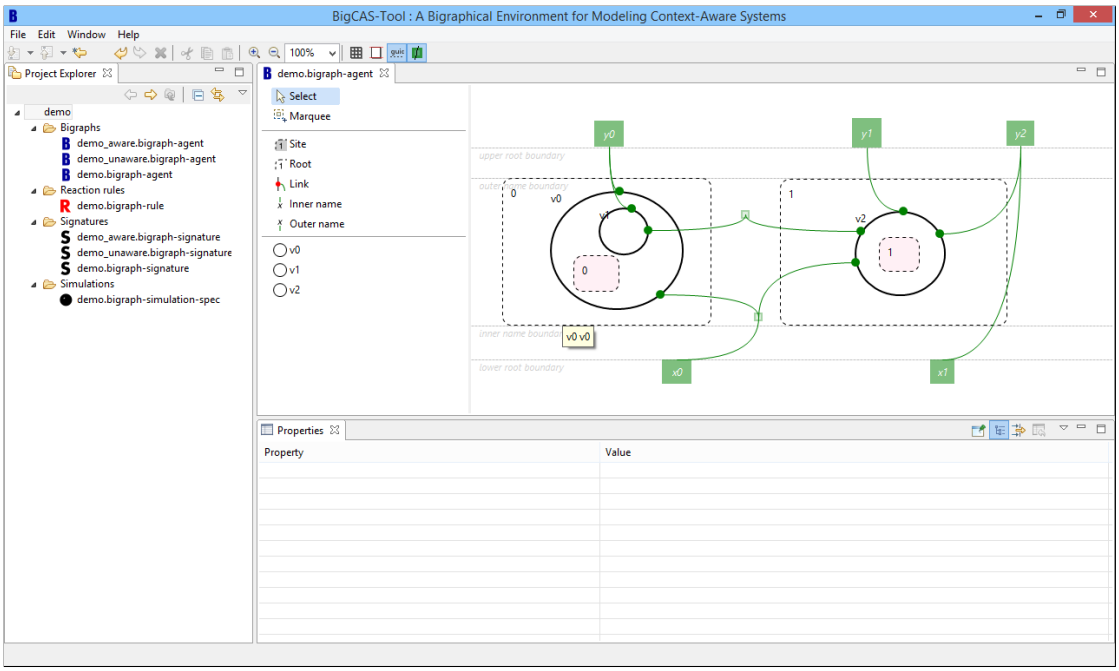


FIGURE 6.3 – Aperçu du prototype BigCAS-Tool

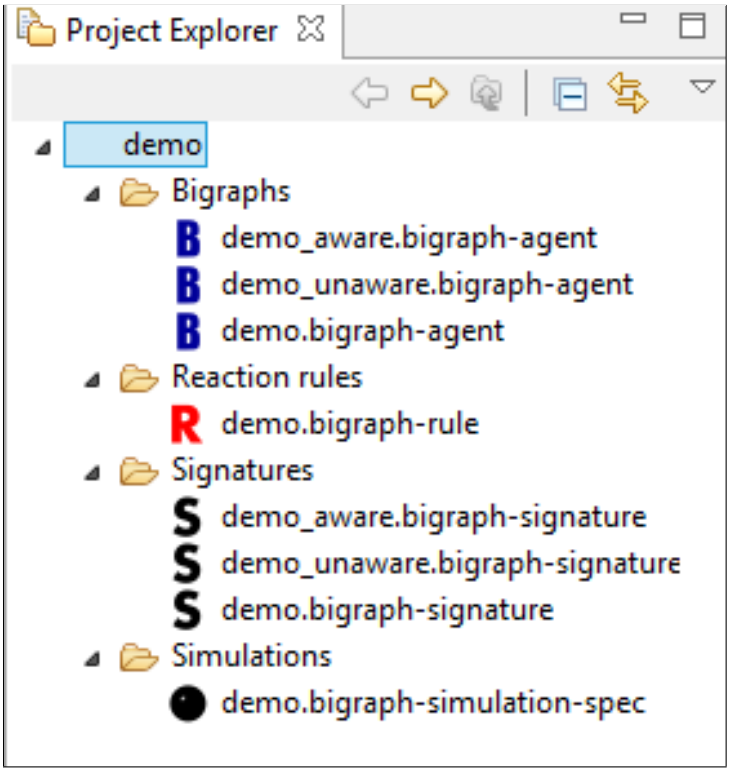


FIGURE 6.4 – Projet BigCAS-Tool

— **Bigraphs :** comprend *_aware.bigraph-agent et *_unaware.bigraph-agent qui sont des fichiers pour gérer les modèles bigraphiques sensibles au contexte et non-sensibles au contexte respectivement.

— **Reaction rules :** contient les fichiers *_.bigraph-rule qui permettent aux utilisateurs de définir les représentations graphiques des règles de réaction, qui peuvent être exportées vers le langage de termes de BigMC [PDH12] pour les exécuter.

— **Signatures :** comprend les fichiers *_aware.bigraph-signature et *_unaware.bigraph-signature qui gèrent séparément et respectivement les nœuds contextuels et les nœuds internes.

— **Simulations :** contient le fichier *_.bigraph-simulation-spec qui fournit une option pour exporter les modèles bigraphiques en forme des documents XML pour effectuer facilement des simulations sur les systèmes.

Éditeur de Bigraphes

En effet, BigCAS-Tool utilise l'éditeur Big Red [FPH13] pour interagir avec les bigraphes. Comme indiqué sur la figure 6.5, la palette sur le côté gauche du canevas permet un accès rapide à l'ensemble des éléments utilisés dans la création et l'édition des modèles bigraphiques, à l'aide de l'opération glisser-déposer (*drag and drop* en anglais). Par exemple, le bouton Root(racine) est utilisé pour créer des nouvelles régions où les nœuds tels que v_0 , v_1 et v_2 peuvent être regroupés, le bouton Link (lien) peut être utilisé pour connecter les ports des différents nœuds, les noms internes (Inner name), et les noms externes (Outer name). La vue sur la droite est le canevas principal où le modèle bigraphique s'affiche.

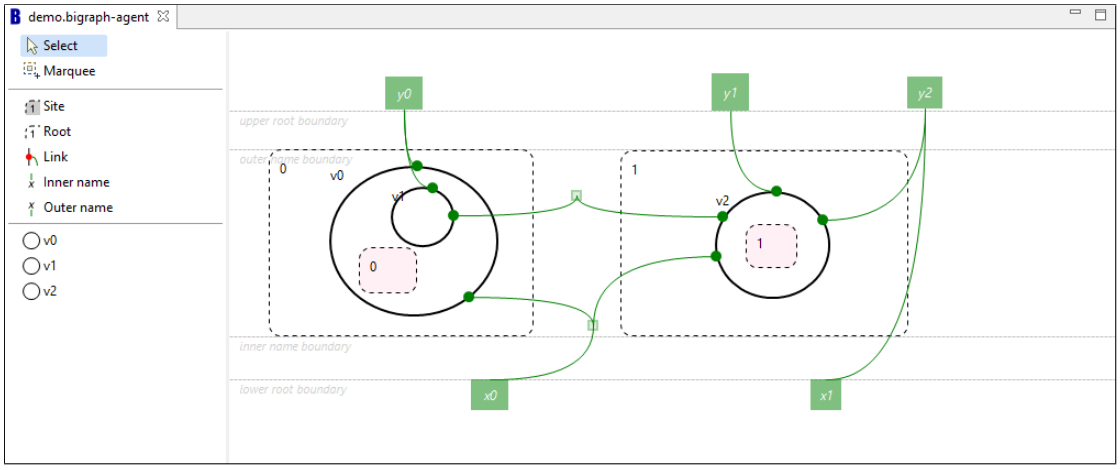


FIGURE 6.5 – Canevas de l'éditeur de bigraphes

Éditeur de Signatures

L'éditeur de signatures (figure 6.6) gère à la fois les propriétés graphiques et les propriétés formelles des nœuds. Il permet de créer (bouton :Add) des nœuds et d'affecter un contrôle à chaque nœud indiquant son identité (Name et Label) ainsi que s'il est atomique (Atomic), actif (Active) ou passif Passive. Les nœuds créés apparaissent dans la partie gauche de l'éditeur. En outre, l'utilisateur peut importer (bouton : Import) ou supprimer (bouton : Remove) une signature déjà créée.

Pour personnaliser l'apparence d'un nœud, le canevas au centre droit permet de sélectionner facilement un dessin préconçu : ovale (Oval) ou polygonale (Polygon), et de choisir une couleur pour le contour (Outiline) et le contenu (Fill). Enfin, les points rouges 1 et 2 désignent les ports où leur nombre définit l'arité du nœud.

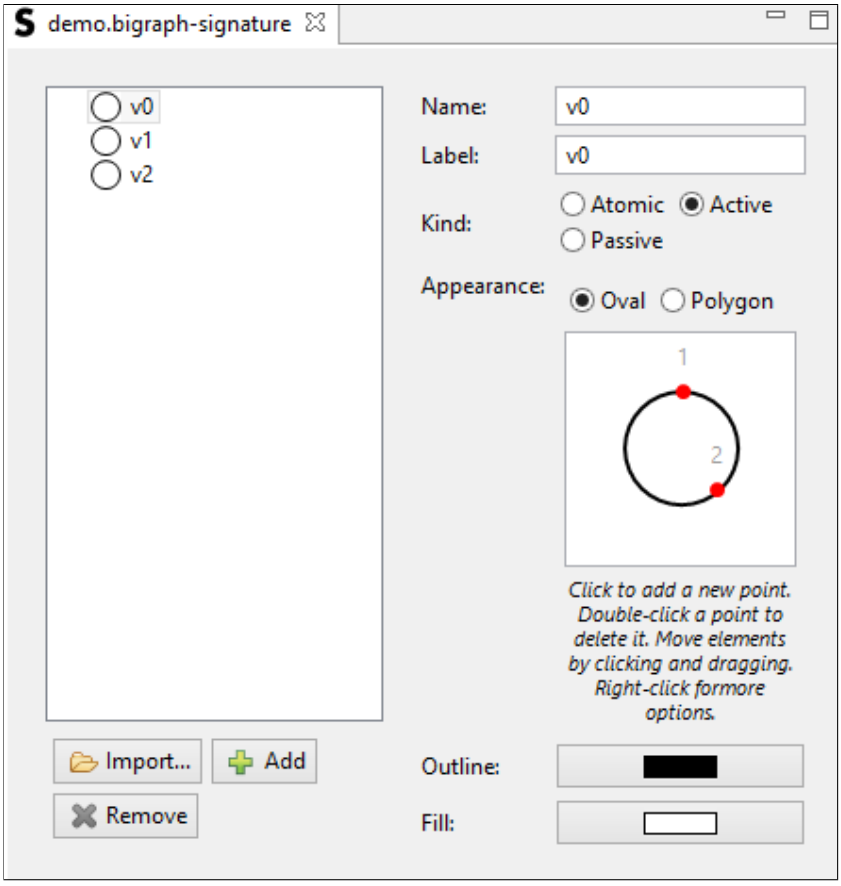


FIGURE 6.6 – Éditeur de signatures

Éditeur de Règles de Réaction

L'éditeur de règles de réaction permet à l'utilisateur de spécifier graphiquement la dynamique des systèmes sensibles au contexte. Comme le montre la figure 6.7, la palette sur la gauche fournit l'ensemble des éléments utilisés pour créer et éditer facilement le redex et le reactum d'une règle de réaction. Le canevas de centre-gauche contient le redex, tandis que le canevas de centre-droit affiche le reactum.

En effet, l'opération de composition est implémentée comme une règle de réaction autogénérée qui a pour objectif la création du modèle de système sensible au contexte en combinant le modèle *_aware.bigraph-agent et le modèle *_unaware.bigraph-agent.

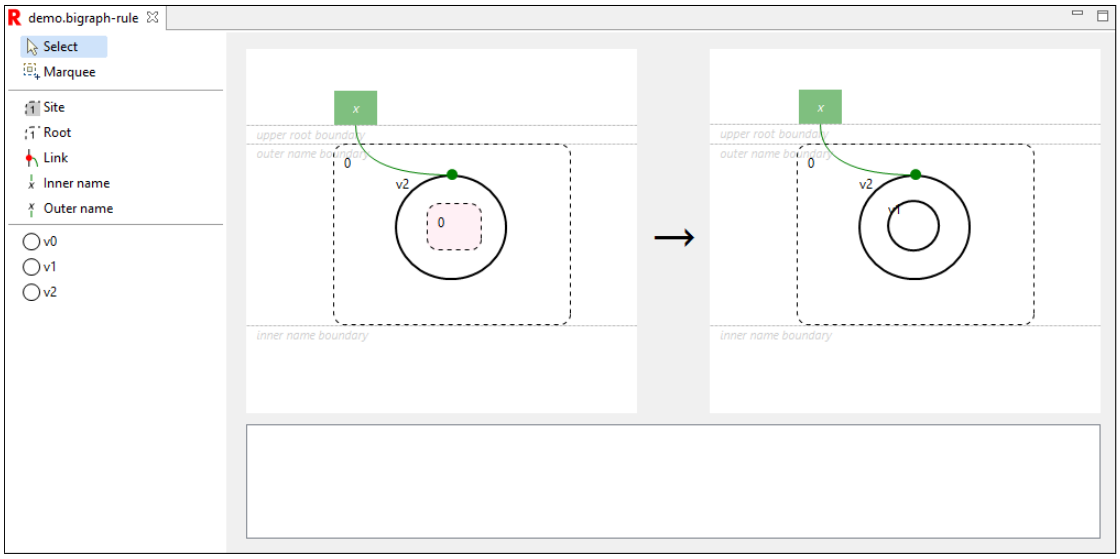


FIGURE 6.7 – Éditeur de règles de réaction

6.5 Accessibilité

Pour permettre l'interfaçage avec les autres outils de bigraphe existants, il est également nécessaire de rendre accessible les modèle des systèmes sensibles au contexte décrits sur BigCAS-Tool. Pour cela, le parseur présenté dans la figure 6.8 fournit un panel qui permet à l'utilisateur d'exporter un modèle bigraphique spécifique dans un fichier XML (figure 6.9), en sélectionnant le modèle, sa signature et un ensemble de règles de réaction, afin d'effectuer des traitements ultérieurs.

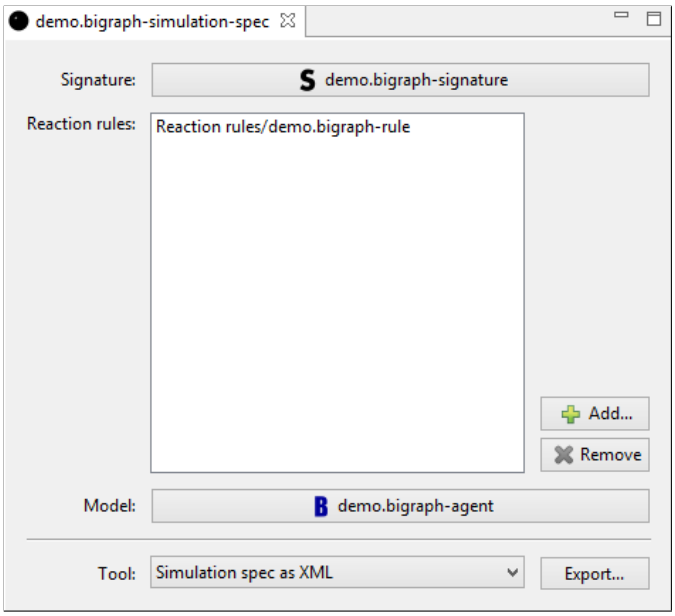


FIGURE 6.8 – Parseur XML

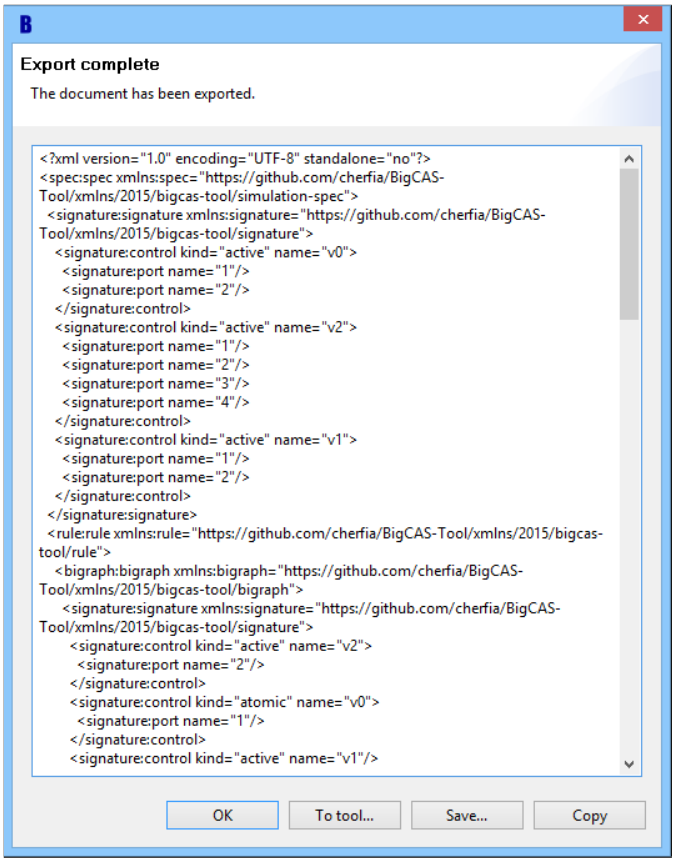


FIGURE 6.9 – Résultat XML sous BigCAS-Tool

6.6 Extensibilité

BigCAS-Tool est une plateforme extensible et ouverte aux nouvelles fonctionnalités avec des modules entièrement personnalisables. Les développeurs peuvent facilement ajouter des fonctionnalités personnalisées et adaptées à leurs besoins, sans introduire de dépendances supplémentaires grâce à un point d'extension. Le point d'extension est un mécanisme, propre à Eclipse, décrivant un contrat défini au travers d'un fichier EXSD (*Extension XML Schema Definition*) (annexe A.2) entre la plateforme BigCAS-Tool et les outils souhaitant s'interfacer avec elle. Par exemple, BigCAS-Tool s'interface aisément avec le model-checker BigMC [PDH12] pour permettre la vérification des modèles de systèmes sensibles au contexte. Pour ce faire, nous avons implémenté un plugin (figure 6.10) permettant d'exporter un modèle BigCAS-FA en langage de termes BigMC, ensuite, ce dernier s'exécute en arrière-plan pour vérifier le modèle. Enfin, le résultat de la vérification s'affiche automatiquement sur une nouvelle fenêtre, comme le montre la figure 6.11.

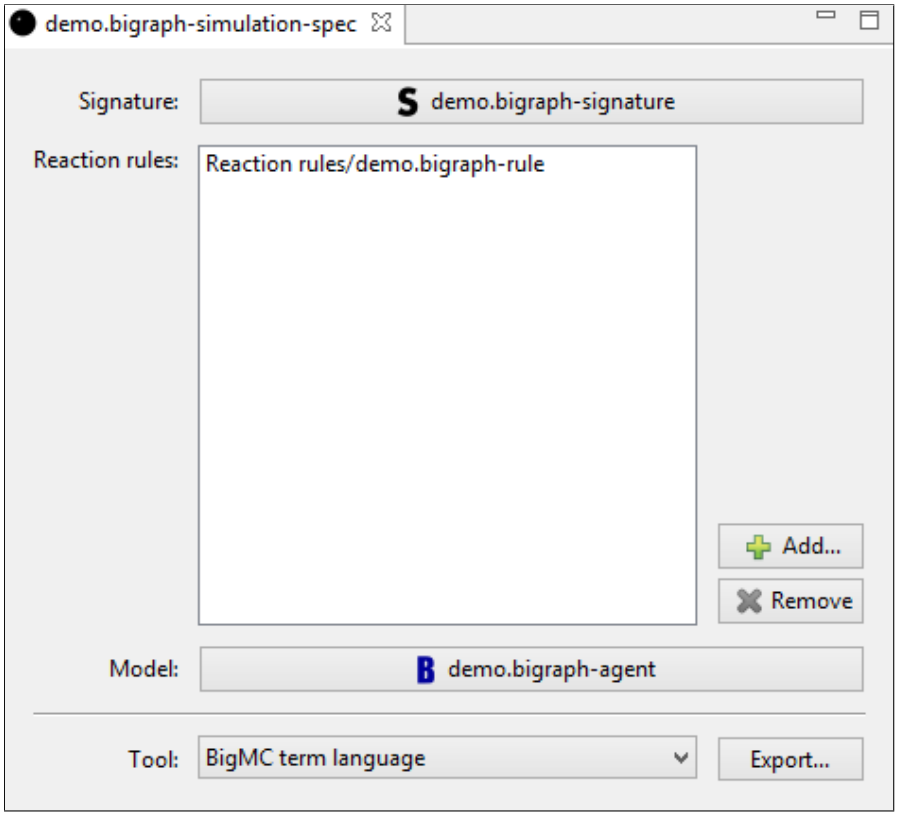


FIGURE 6.10 – Point d'extension BigMC

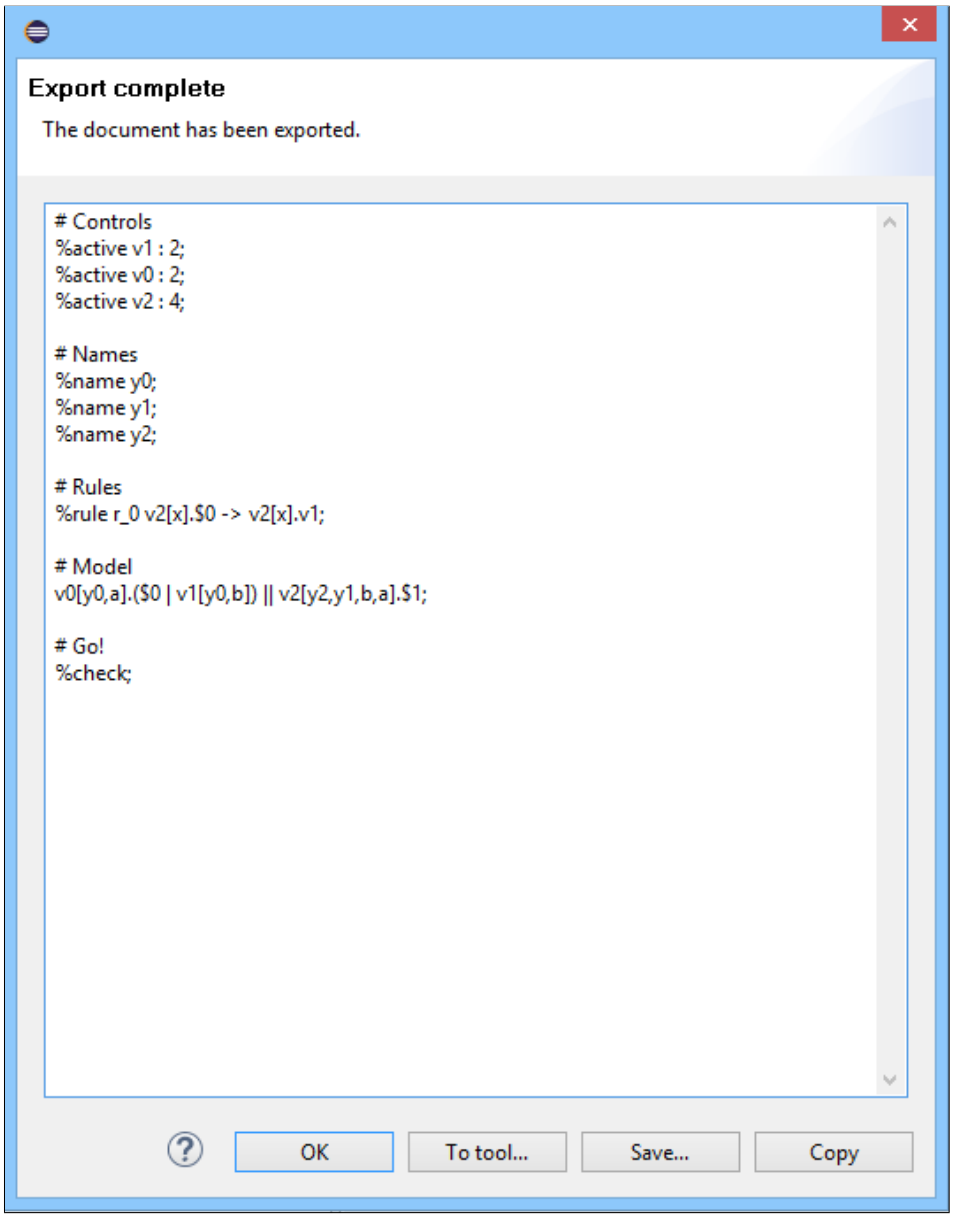


FIGURE 6.11 – Code BigMC généré

6.7 Expérimentation

Dans cette section, nous présentons le principe de fonctionnement de BigCAS-Tool à travers l'étude de cas de système d'éclairage intelligent présentée dans le chapitre 4 (section 4.3.1).

La figure 6.12 donne un aperçu du système d'éclairage intelligent sous la plateforme BigCAS-Tool.

FIGURE 6.12 – Aperçu de projet du système d'éclairage intelligent

La figure 6.13 représente le projet BigCAS-Tool du système d'éclairage intelligent. Le projet

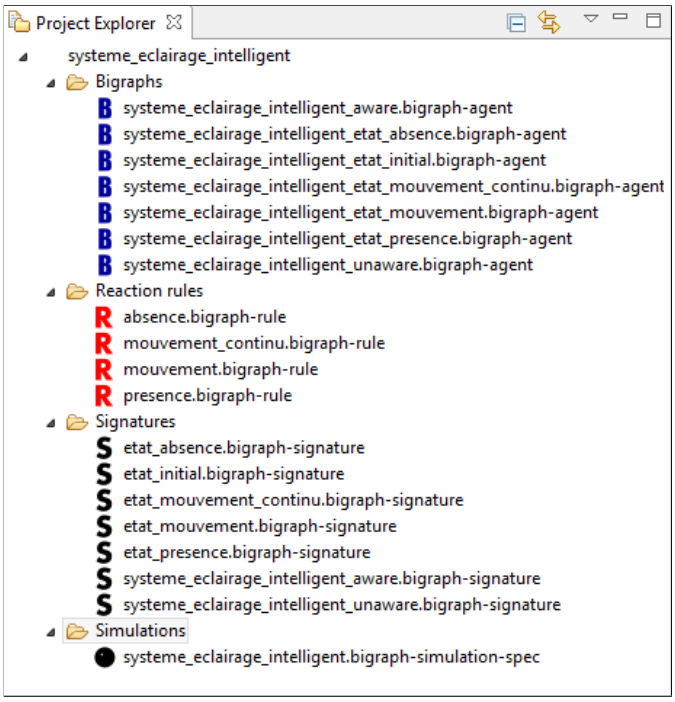


FIGURE 6.13 – Arborescence du projet système d'éclairage intelligent

systeme_eclairage_intelligent contient les fichiers suivants :

— **Bigraphes :**

systeme_eclairage_intelligent_aware.bigraph-agent : modèle de bigraphe sensible au contexte (figure 6.17).

systeme_eclairage_intelligent_etat_absence.bigraph-agent : modèle de bigraphe de l'état d'absence (figure 6.22).

systeme_eclairage_intelligent_etat_initial.bigraph-agent : modèle de bigraphe de l'état initial (figure 6.15).

systeme_eclairage_intelligent_etat_mouvement_continu.bigraph-agent : modèle de bigraphe de l'état de mouvement continu (figure 6.21).

systeme_eclairage_intelligent_etat_mouvement.bigraph-agent : modèle de bigraphe de l'état de mouvement (figure 6.20).

systeme_eclairage_intelligent_etat_presence.bigraph-agent : modèle de bigraphe de l'état de présence (figure 6.18).

systeme_eclairage_intelligent_unaware.bigraph-agent : modèle de bigraphe non-sensible au contexte (figure 6.16).

— **Règles de réactions :**

absence.bigraph-rule : règle de réaction de l'état d'absence.

mouvement_continu.bigraph-rule : règle de réaction de l'état de mouvement continu.

mouvement.bigraph-rule : règle de réaction de l'état de mouvement.

presence.bigraph-rule : règle de réaction de l'état de présence.

— **Signatures :**

etat_absence.bigraph-signature : signature de bigraphe de l'état d'absence.

etat_initial.bigraph-signature : signature de bigraphe de l'état initial (figure 6.14).

etat_mouvement_continu.bigraph-signature : signature de bigraphe de l'état de mouvement continu.

etat_mouvement.bigraph-signature : signature de bigraphe de l'état de l'état de mouvement.

etat_presence.bigraph-signature signature de bigraphe de l'état de présence.

systeme_eclairage_intelligent_aware.bigraph-signature : signature de bigraphe sensible au contexte.

systeme_eclairage_intelligent_unaware.bigraph-signature : signature de bigraphe non-sensible au contexte.

— **Simulations :**

systeme_eclairage_intelligent.bigraph-simulation-spec : fichier correspondant aux différentes simulations à réaliser sur le projet systeme_eclairage_intelligent.

Pour effectuer l'analyse des propriétés fonctionnelles du modèle de système d'éclairage intelligent, BigCAS-Tool fournit un support permettant l'exportation de ce modèle en langage de termes BigMC comme montré dans la figure 6.23.

6.8 Conclusion

Dans ce chapitre, nous avons présenté un prototype, appelé BigCAS-Tool (Bigraphical Context Aware System - Tool), dédié à la modélisation et à la vérification des systèmes sensibles au contexte. BigCAS-Tool est une extension de l'outil Big Red [FPH13] qui fournit un ensemble d'éditeurs graphiques permettant la manipulation des systèmes sensibles au contexte exprimés sous forme de bigraphe. De plus, ce prototype est conçu pour être facilement utilisé sans une connaissance approfondie des caractéristiques de systèmes réactifs bigraphiques. En effet, la seule tâche à la charge de l'utilisateur est celle de la description du modèle de système.

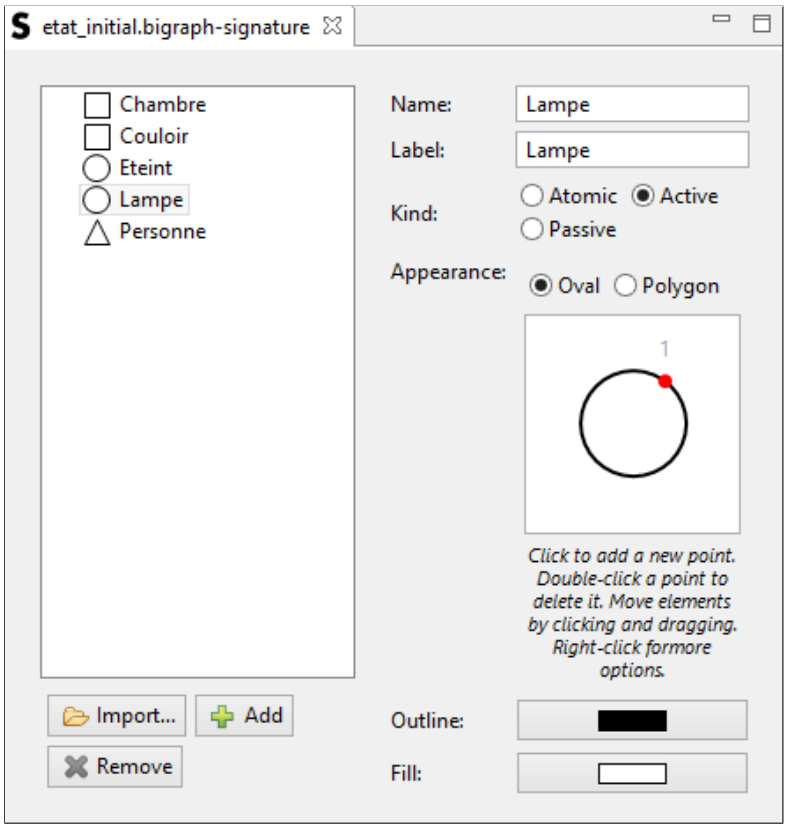


FIGURE 6.14 – Signature de l'état initial

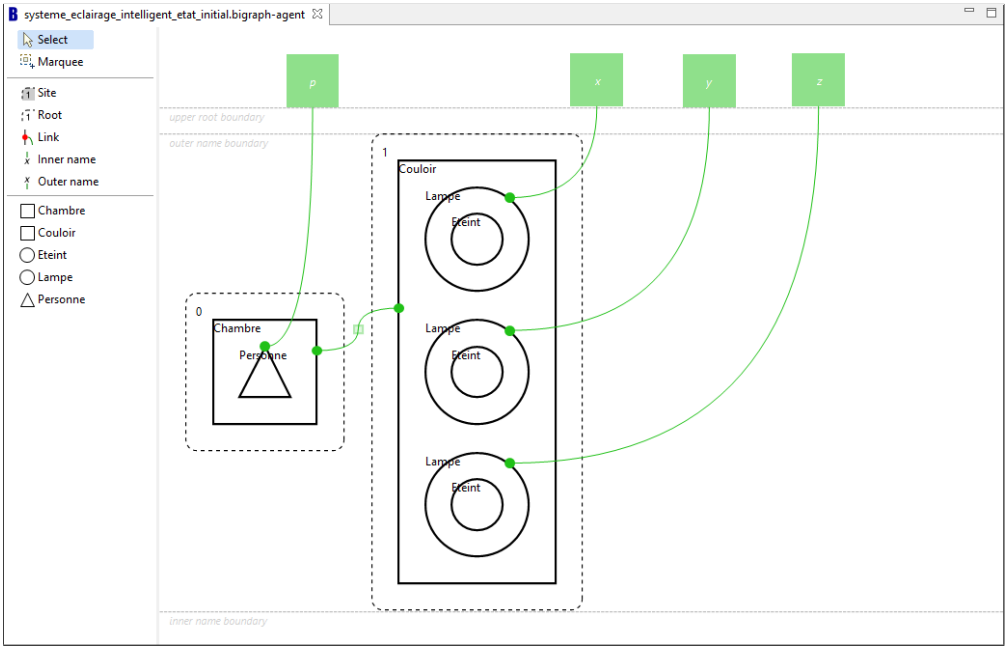


FIGURE 6.15 – Bigraphe d'état initial

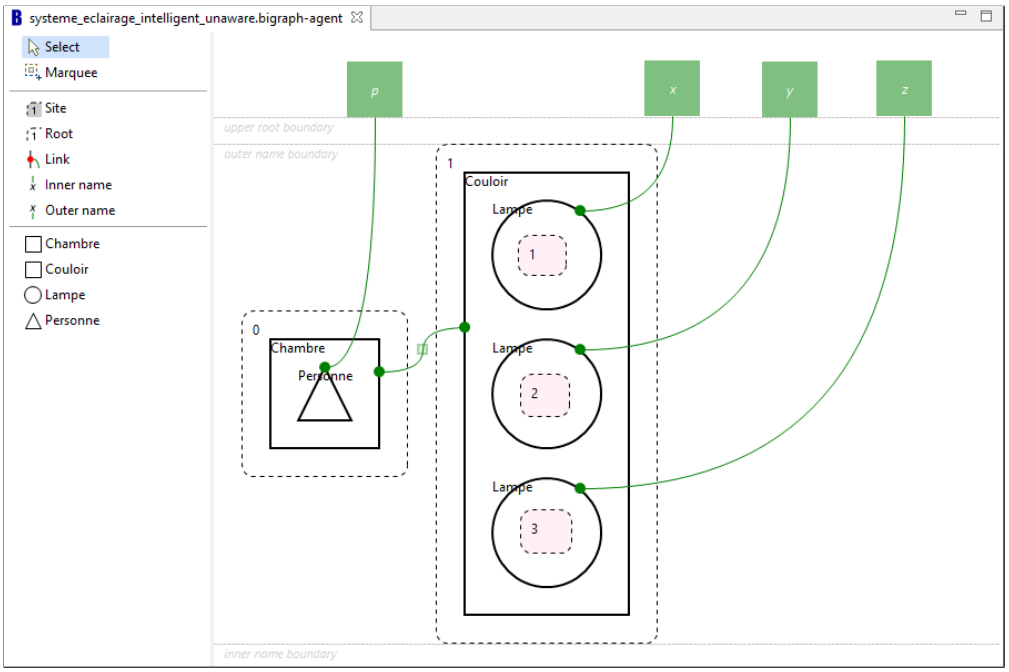


FIGURE 6.16 – Bigraphe non-sensible au contexte de l'état de présence

FIGURE 6.17 – Bigraphe sensible contexte de l'état de présence

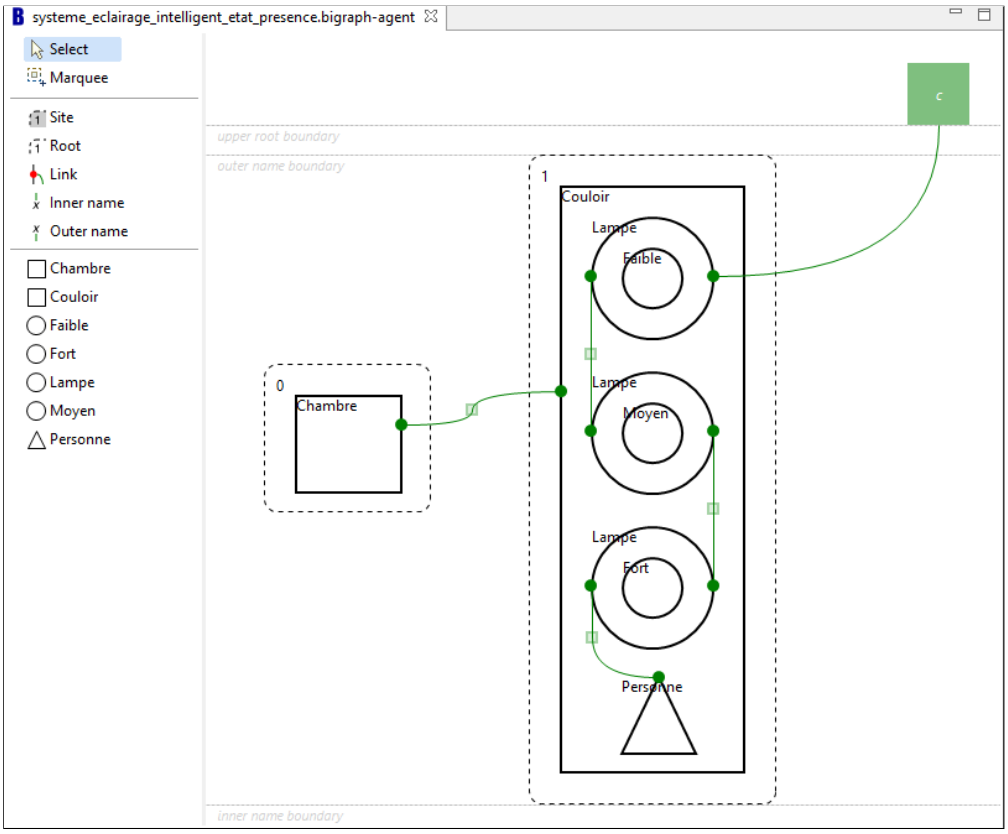


FIGURE 6.18 – Bigraphe de l'état de présence

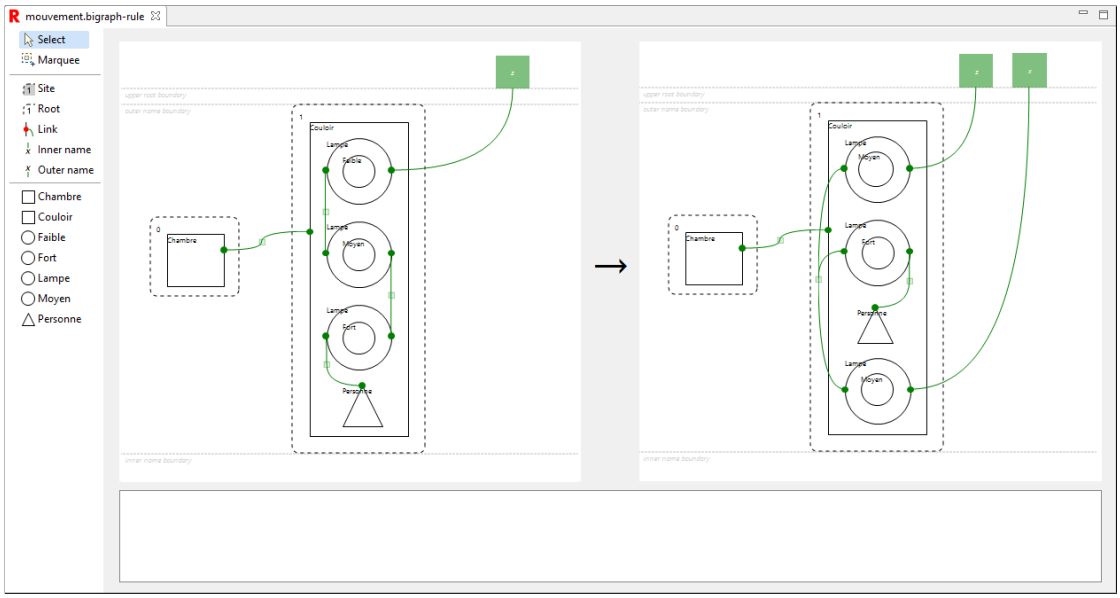


FIGURE 6.19 – Exemple de règle de réaction de l'état de mouvement sous BigCAS-Tool

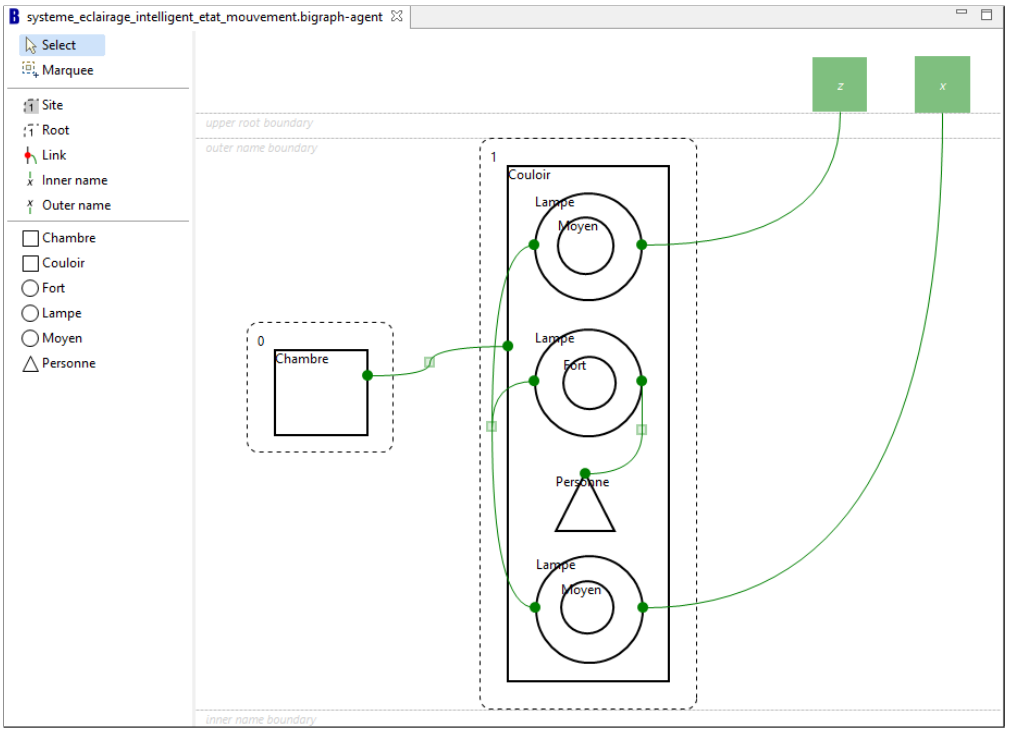


FIGURE 6.20 – Bigraphe de l'état de mouvement

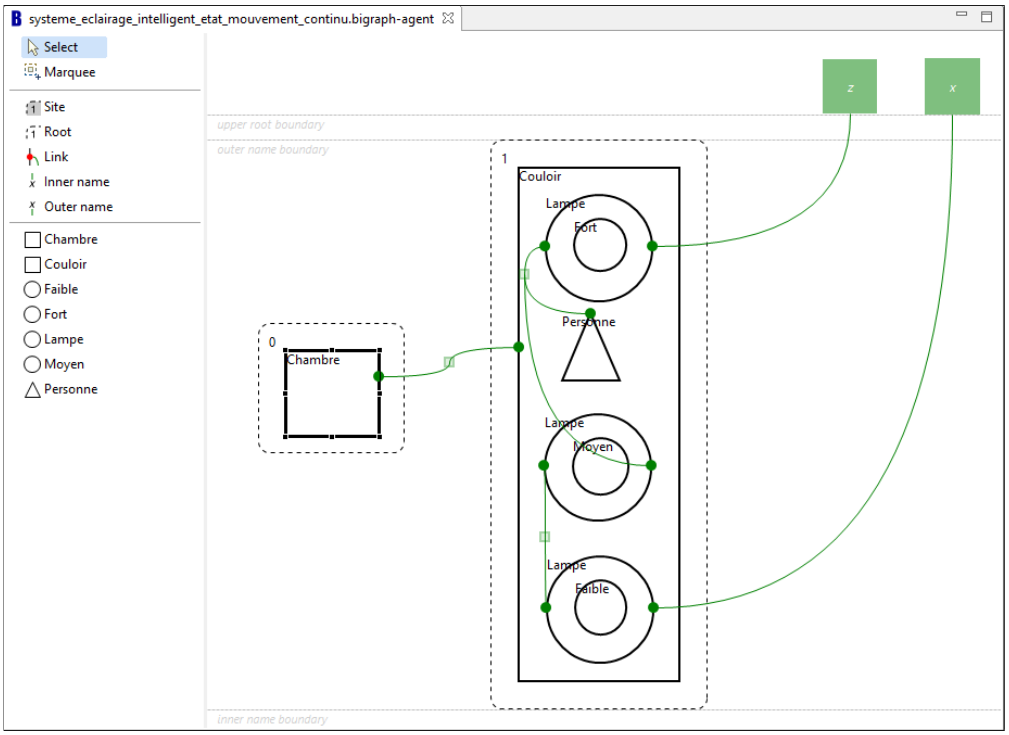


FIGURE 6.21 – Bigraphe de l'état de mouvement continu

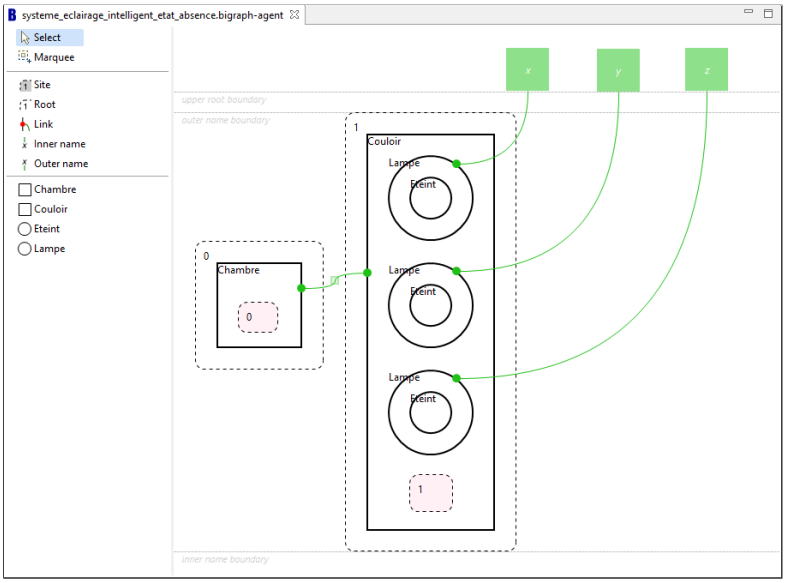


FIGURE 6.22 – Bigraphe de l'état d'absence

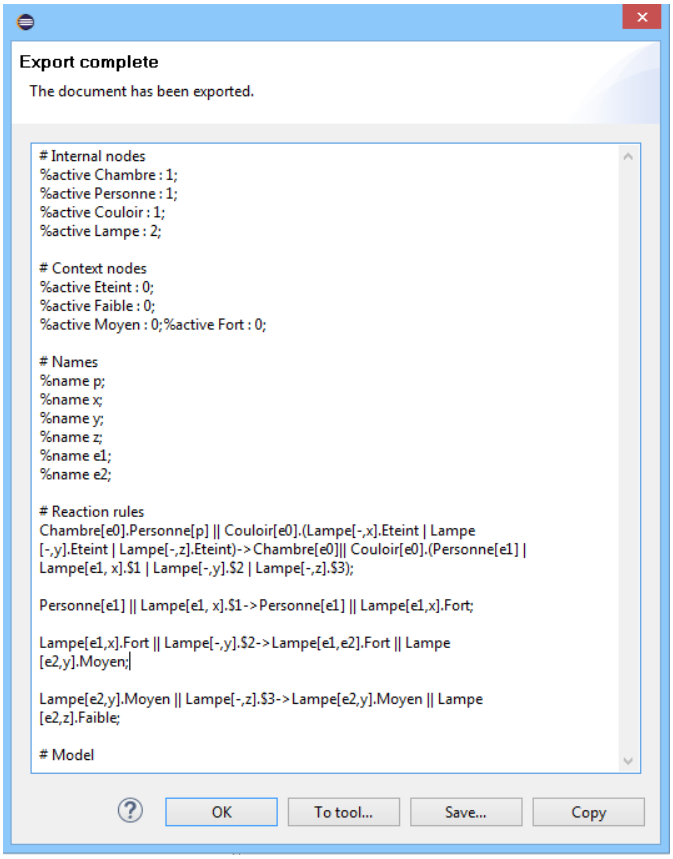


FIGURE 6.23 – Exportation du modèle de système d'éclairage intelligent en langage de termes BigMC

Conclusion Générale

Le travail réalisé dans le cadre de cette thèse se positionne dans le domaine de l'informatique sensible au contexte qui constitue la pierre angulaire de l'informatique future. L'informatique sensible au contexte regroupe les aspects de la dynamicité, la distribution et la mobilité, et adopte une nouvelle vision qui met l'information toujours à disposition de l'utilisateur sans explicitement attirer son attention. La sensibilité au contexte est la propriété principale dans ce domaine. Elle caractérise la capacité d'un système informatique de s'adapter facilement aux changements de l'environnement et aux besoins de l'utilisateur.

Cependant, la compréhension du contexte par un système informatique impose plusieurs défis dans l'informatique sensible au contexte. En effet, les informations de contexte sont généralement des informations brutes collectées à partir de sources hétérogènes. De ce fait, elles peuvent être ambigües, incohérentes ou même incomplètes. Les informations de contexte peuvent également varier relativement des données simples aux complexes qui ne sont pas appropriées pour une utilisation directe par le système, ce qui rend leur interprétation particulièrement difficile. Pour que le système informatique puisse comprendre les informations de contexte, il est nécessaire de fournir un niveau d'abstraction à ces informations en mettant en place un modèle qui permet de fournir une description abstraite des informations contextuelles. Ce modèle doit être générique, c'est-à-dire, qu'il peut être utilisé et instanciable dans différents domaines.

L'objectif de cette thèse étant de définir un modèle à base des systèmes réactifs bigraphiques pour la spécification et la vérification formelles des systèmes sensibles au contexte. Nous avons entamé notre étude par un état de l'art sur l'informatique sensible au contexte. Tout d'abord, nous avons présenté les différentes définitions du contexte proposées dans la littérature, ses caractéristiques et l'importance de sa prise en compte dans le domaine informatique. Puis, nous avons introduit les différentes fonctionnalités des systèmes sensibles au contexte, à savoir l'acquisition, la représentation et l'adaptation au contexte. Ensuite, nous avons étudié différentes approches de modélisation des systèmes sensibles au contexte et l'apport de chacune concernant l'expressivité et l'abstraction du modèle, la possibilité de sa réutilisation et son extension, la spécificité du raisonnement et la possibilité de partager

les informations contextuelles. Cette étude nous a permis, d'une part, de mieux cerner les difficultés liées à la modélisation des systèmes sensibles au contexte, et d'autre part, de choisir le modèle des systèmes réactifs bigraphiques qui nous semble être le mieux adapté à la description du contexte dans un environnement supportant les caractéristiques ubiquitaires. Cependant, les approches à base des systèmes réactifs bigraphiques existantes ne prennent en compte que l'aspect statique lié à la structure alors que l'essence même des systèmes sensibles au contexte est dynamique.

Pour palier à ce manque, nous avons proposé un modèle à base des systèmes réactifs bigraphiques pour modéliser les aspects statiques et dynamiques des systèmes sensibles au contexte. Tout d'abord, nous avons présenté une revue de littérature sur les systèmes réactifs bigraphiques. Nous avons commencé par décrire l'anatomie des bigraphes, leurs caractéristiques et les principales opérations que nous pouvons effectuer sur eux. Puis, nous avons introduit la dynamique de ce formalisme ainsi que leur langage de termes algébrique. Enfin, nous avons présenté les principaux outils développés autour des systèmes réactifs bigraphiques.

Dans cette thèse, nous nous sommes intéressés à la modélisation des systèmes sensibles au contexte, en prenant en compte tous les éléments du contexte, ainsi que les aspects statiques et dynamiques qui les concernent. Nos contributions s'étendent de la spécification à la mise en œuvre d'un outil pour la sensibilité au contexte. Nous résumons ces contributions en trois points essentiels.

BigCAS : Modèle de Spécification des Systèmes Sensibles au Contexte

Nous avons défini un modèle, dit BigCAS (Bigraphical Context-Aware System), à base des systèmes réactifs bigraphiques qui permet de décrire les aspects structurels et comportementaux des systèmes sensibles au contexte. La structure de BigCAS comprend deux bigraphes distincts, un bigraphe non-sensible au contexte permettant de modéliser la partie du système dont la structure reste constante, même si le contexte qui entoure le système change, alors que le bigraphe sensible au contexte permet de décrire la partie du système dont la structure supporte certaine variabilité en fonction d'éventuels changements du contexte. La composition de ces deux bigraphes permet de décrire la structure générale d'un système sensible au contexte. La valeur ajoutée du modèle BigCAS, par rapport aux approches orientées systèmes réactifs bigraphiques citées dans le chapitre 2, réside dans le fait qu'il permet non seulement la description de l'aspect structurel des systèmes sensibles au contexte, mais il permet aussi la modélisation des différentes reconfigurations internes et contextuelles intervenant dans le comportement de tels systèmes.

BigCAS-FA : Extension pour l'Analyse des Systèmes Sensibles au Contexte

Nous avons proposé une extension de notre modèle, appelée BigCAS-FA (Bigraphical Context-Aware System - Formal Analysis), permettant l'analyse formelle des systèmes sensibles au contexte. BigCAS-FA permet de vérifier certaines propriétés relatives à la sûreté et à la vivacité des systèmes sensibles au contexte. En outre, nous avons défini des stratégies de reconfiguration à base de contrats qui consistent principalement en un ensemble de règles de réaction bigraphiques déclenchées par des événements conduisant à une adaptation sensible au contexte. Les stratégies de reconfiguration proposées ont été prises en charge par BigCAS-FA pour vérifier les propriétés relatives aux contrats d'adaptation.

BigCAS-Tool : Plateforme pour la manipulation des Systèmes Sensibles au Contexte

Afin de valider nos contributions théoriques, nous avons développé un prototype, appelé BigCAS-Tool (Bigraphical Context Aware System - Tool), permettant la manipulation des systèmes sensibles au contexte. Ce prototype est une extension de l'outil Big Red [FPH13] qui consiste en un ensemble d'éditeurs graphiques pour la modélisation et la vérification des modèles de bigraphes associés aux systèmes sensibles au contexte. En effet, BigCAS-Tool prend en compte tous les éléments fonctionnels d'un système sensible au contexte, la seule tâche à la charge de l'utilisateur est celle de la description du modèle de système.

Les travaux que nous avons proposé ouvrent plusieurs perspectives scientifiques à court et à plus long terme. Nous soulignons ici certaines de ces perspectives.

— Nous visons à étendre le modèle BigCAS pour prendre en compte l'aspect temporel afin de caractériser l'évolution des systèmes sensibles au contexte.

— BigCAS prend en compte la structure et le comportement dynamique des systèmes sensible au contexte. Il nous semble également intéressant d'étendre BigCAS pour traiter d'autres aspects tels que : les aspects adaptatifs et les aspects cognitifs.

— Il nous semble aussi très intéressant d'intégrer BigCAS-FA à la plateforme Maude ce qui nous permettrait de vérifier d'autres propriétés telle que la qualité de contexte (QoC).

— Actuellement, le prototype BigCAS-Tool a été testé sur des cas simples, mais représentatifs de certains scénarios de sensibilité au contexte réels. Nous estimons qu'il est important de tester son fonctionnement sur un cas concret d'un système sensible au contexte.

— Nous envisageons d'implémenter les fonctionnalités manquantes dans le prototype actuel, telles que les stratégies de reconfiguration à base de contrats.

— Une autre piste pour améliorer BigCAS-Tool serait de supporter le raisonnement temporel sur les évolutions des systèmes sensibles au contexte.

Bibliographie

[AD94] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2) :183–235, 1994.

[ADB+99] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[Agh86] Gul Agha. *An overview of actor languages*, volume 21. ACM, 1986.

[Ald03] Frances K Aldrich. Smart homes : past, present and future. In *Inside the smart home*, pages 17–39. Springer, 2003.

[AVF07] Karine Abbas, Christine Verdier, and André Flory. Exploiting profile modeling for web-based information systems. In *Web Information Systems Engineering–WISE 2007 Workshops*, pages 313–324. Springer, 2007.

[Bar03] Louise Barkhuus. Context information vs. sensor information : A model for categorizing context in context-aware mobile computing. *SIMULATION SERIES*, 35(1) :127–133, 2003.

[BB89] Gérard Berry and Gérard Boudol. The chemical abstract machine. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 81–94. ACM, 1989.

[BBC97] Peter J Brown, John D Bovey, and Xian Chen. Context-aware applications : from the laboratory to the marketplace. *Personal Communications, IEEE*, 4(5) :58–64, 1997.

[BBH+10] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2) :161–180, 2010.

[BDE+06] Lars Birkedal, Søren Debois, Ebbe Elsborg, Thomas Hildebrandt, and Henning Niss. Bigraphical models of context-aware systems. In *Foundations of software science and computation structures*, pages 187–201. Springer, 2006.

[BDGM07] Lars Birkedal, Troels Christoffer Damgaard, Arne John Glenstrup, and Robin Milner. Matching of bigraphs. *Electronic Notes in Theoretical Computer Science*, 175(4) :3–19, 2007.

[BDR07] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4) :263–277, 2007.

[BEH+09] A Badii, I Etxeberria, C Huijnen, M Maseda, S Dittenberger, A Hochgatterer, D Thiemert, and AS Rigaud. Companionable : Graceful integration of mobile robot companion with a smart home environment. *Gerontechnology*, 8(3) :181, 2009.

[Ben07] Jørgen Eske Runge Bentzen. the aran protocol. Master's thesis, IT University of Copenhagen, 2007.

[Ber99] Cathy Berthouzoz. A model of context adapted to domain-independent machine translation. In *Modeling and Using Context*, pages 54–66. Springer, 1999.

[Ber08] Gérard Berry. *Pourquoi et comment le monde devient numérique*. Collège de France, 2008.

[BGH+12] Mikkel Bundgaard, Arne J Glenstrup, Thomas Hildebrandt, Espen Højsgaard, and Henning Niss. Formalizing ws-bpel and higher order mobile embedded business processes in the bigraphical programming languages (bpl) tool. *Bigraphical Languages and their Simulation*, page 115, 2012.

[BHH04] Sven Buchholz, Thomas Hamann, and Gerald Hubsch. Comprehensive structured context profiles (cscp) : Design and experiences. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 43–47. IEEE, 2004.

[BJ01] Peter J Brown and Gareth JF Jones. Context-aware retrieval : Exploring a new environment for information retrieval and information filtering. *Personal and Ubiquitous Computing*, 5(4) :253–263, 2001.

[BJPW99] Antoine Beugnard, Jean-Marc Jézéquel, Noël Plouzeau, and Damien Watkins. Making components contract aware. *Computer*, 32(7) :38–45, 1999.

[BK86] JA Bergstra and Jan Willem Klop. Algebra of communicating processes. *Mathematics and Computer Science, CWI Monograph*, 1 :89–138, 1986.

[BKLPS04] Thomas Buchholz, Michael Krause, Claudia Linnhoff-Popien, and Michael Schiffers. Coco : dynamic composition of context information. In *Mobile and Ubiquitous Systems : Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 335–343. IEEE, 2004.

[BKS03] Thomas Buchholz, Axel Küpper, and Michael Schiffers. Quality of context : What it is and why we need it. In *Proceedings of the workshop of the HP OpenView University Association*, volume 2003, 2003.

[Bré02] Patrick Brézillon. Hors du contexte, point de salut. *Séminaire" Objets Communicants*, 2002.

[Bro95] Peter J Brown. The stick-e document : a framework for creating context-aware applications. *ELECTRONIC PUBLISHING-CHICHESTER-*, 8 :259–272, 1995.

[BS97] Marko Balabanović and Yoav Shoham. Fab : content-based, collaborative recommendation. *Communications of the ACM*, 40(3) :66–72, 1997.

[BSO07] Gautier Bastide, Abdelhak Seriai, and Mourad Oussalah. Software component re-engineering for their runtime structural adaptation. In *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, volume 1, pages 109–114. IEEE, 2007.

[CB13] Taha Abdelmoutaleb Cherfia and Faïza Belala. Towards a bigraph-based model for context-aware adaptive systems. In *Software Architecture*, pages 340–343. Springer, 2013.

[CB14] Taha Abdelmoutaleb Cherfia and Faïza Belala. Bigraphical reactive systems based approaches for modeling context-aware systems. *International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS)*, 5(4) :1–19, 2014.

[CBB14a] Taha Abdelmoutaleb Cherfia, Kamel Barkaoui, and Faïza Belala. A brs-based modeling approach for context-aware systems : A case study of smart car system. In *Embedded and Ubiquitous Computing (EUC), 2014 12th IEEE International Conference on*, pages 310–314. IEEE, 2014.

[CBB14b] Taha Abdelmoutaleb Cherfia, Faiza Belala, and Kamel Barkaoui. Towards formal modeling and verification of context-aware systems. In *Proceedings of the 8th International Workshop on Verification and Evaluation of Computer and Communication Systems, VECoS 2014, Bejaïa, Algeria, September 29-30, 2014.*, pages 18–24, 2014.

[CCDG05] Joëlle Coutaz, James L Crowley, Simon Dobson, and David Garlan. Context is key. *Communications of the ACM*, 48(3) :49–53, 2005.

[CCdL+09] Tiziana Catarci, Febo Cincotti, Massimiliano de Leoni, Massimo Mecella, and Giuseppe Santucci. Smart homes for all : Collaborating services in a for-all architecture for domotics. In *Collaborative Computing : Networking, Applications and Worksharing*, pages 56–69. Springer, 2009.

[CCTK13] Diane J Cook, Aaron S Crandall, Brian L Thomas, and Narayanan C Krishnan. Casas : A smart home in a box. *Computer*, 46(7), 2013.

[CEEC08] Marie Chan, Daniel Estève, Christophe Escriba, and Eric Campo. A review of smart homes - present state and future challenges. *Computer methods and programs in biomedicine*, 91(1) :55–81, 2008.

[CFB12] Taha A Cherfia, Belala Faïza, and Nadira Benlahrache. Modeling of architectural reconfiguration case study : Automated teller machine. In *Advanced Information Systems for Enterprises (IWAISE), 2012 Second International Workshop on*, pages 40–46. IEEE, 2012.

[CFJ03] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review*, 18(03) :197–207, 2003.

[CFJ04a] H Chen, T Finin, and A Joshi. Semantic web in the context broker architecture. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 277–286. IEEE, 2004.

[CFJ04b] Harry Chen, Tim Finin, and Anupam Joshi. *A context broker for building smart meeting rooms*. Defense Technical Information Center, 2004.

[CFJ05] Harry Chen, Tim Finin, and Anupam Joshi. Using owl in a pervasive computing broker. Technical report, DTIC Document, 2005.

[CG98] Luca Cardelli and Andrew D Gordon. Mobile ambients. In *Foundations of Software Science and Computation Structures*, pages 140–155. Springer, 1998.

[CG00] Luca Cardelli and Andrew D Gordon. Mobile ambients. *Theoretical computer science*, 240(1) :177–213, 2000.

[CGD+06] Christophe Chassot, Karim Guennoun, Khalil Drira, Francois Armando, Ernesto Exposito, and André Lozes. Towards autonomous management of qos through model-driven adaptability in communication-centric systems. *ITSSA*, 2(3) :255–264, 2006.

[Cha07] Tarak Chaari. *Adaptation d'applications pervasives dans des environnements multi-contextes*. PhD thesis, INSA de Lyon, 2007.

[Che04] Harry Lik Chen. *An intelligent broker architecture for pervasive context-aware systems*. PhD thesis, University of Maryland, Baltimore County, 2004.

[CK+00] Guanling Chen, David Kotz, et al. A survey of context-aware mobile computing research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

[CLF05] Tarak Chaari, Frédérique Laforest, and André Flory. Adaptation des applications au contexte en utilisant les services web. In *Proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing*, pages 111–118. ACM, 2005.

[CMD02] Keith Cheverst, Keith Mitchell, and Nigel Davies. The role of adaptive hypermedia in a context-aware tourist guide. *Communications of the ACM*, 45(5) :47–51, 2002.

[CY07] OkJoo Choi and YongIk Yoon. A meta data model of context information for dynamic service adaptation on user centric environment. In *Multimedia and Ubiquitous Engineering, 2007. MUE'07. International Conference on*, pages 108–113. IEEE, 2007.

[Dav05] Pierre-Charles David. *Développement de composants Fractal adaptatifs : un langage dédié à l'aspect d'adaptation*. PhD thesis, Université de Nantes, 2005.

[DD03] I Demeure and A Duda. Systèmes répartis et réseaux adaptifs au contexte (context-aware). *Disponible à l'adresse : http ://www. laas. fr/RTP01-RdC/AS150-AdaptContexte. pdf (Consulté le 01/09/2008)*, 2003.

[Dey01] Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1) :4–7, 2001.

[EGK+02] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C North, and Gordon Woodhull. Graphviz - open source graph drawing tools. In *Graph Drawing*, pages 483–484. Springer, 2002.

[Els09] Ebbe Elsborg. *Bigraphs : Modelling, Simulation, and Type Systems*. PhD thesis, IT-Universitetet i KøbenhavnIT University of Copenhagen, DirektionenManagement, InstituttetThe Department, Programming, Logic and SemanticsProgramming, Logic and Semantics, 2009.

[Ens01] Oliver Enseling. icontract : Design by contract in java. *JavaWorld, May*, 2001.

[ES12] N Eén and N Sörensson. Minisat : A minimalistic and high-performance sat solver, 2012.

[FDCP+05] Michael Friedewald, Olivier Da Costa, Yves Punie, Petteri Alahuhta, and Sirkka Heinonen. Perspectives of ambient intelligence in the home environment. *Telematics and informatics*, 22(3) :221–238, 2005.

[Flo67] Robert W Floyd. Assigning meanings to programs. In *Proc. Symp. in Applied Mathematics*, volume 19, pages 19–32, 1967.

[FPH13] Alexander Faithfull, Gian Perrone, and Thomas T Hildebrandt. Big red : A development environment for bigraphs. In *Selected Revised Papers from the 4th International Workshop on Graph Computation Models (GCM 2012)*, volume 61, 2013.

[GCH+04] David Garlan, Shang-Wen Cheng, An-Cheng Huang, Bradley Schmerl, and Peter Steenkiste. Rainbow : Architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10) :46–54, 2004.

[GDBH11] Arne J Glenstrup, Troels C Damgaard, Lars Birkedal, and Espen Højsgaard. An implementation of bigraph matching. *Bigraphical Languages and their Simulation*, page 55, 2011.

[GLM] Chris Greenhalgh, Brian Logan, and Neil Madden. Bigraphspace : authoring ubiquitous experiences with bigraphs.

[GLT08] Ian Gorton, Yan Liu, and Nihar Trivedi. An extensible and lightweight architecture for adaptive server applications. *Software : Practice and Experience*, 38(8) :853–883, 2008.

[Gru93] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2) :199–220, 1993.

[Gwi00] Jacek Gwizdka. What's in the context. In *Computer Human Interaction*, volume 2000, 2000.

[GWPZ04] Tao Gu, Xiao Hang Wang, Hung Keng Pung, and Da Qing Zhang. An ontology-based context model in intelligent environments. In *Proceedings of communication networks and distributed systems modeling and simulation conference*, volume 2004, pages 270–275, 2004.

[H+85] Charles Antony Richard Hoare et al. *Communicating sequential processes*, volume 178. Prentice-hall Englewood Cliffs, 1985.

[Hal09] Terry Halpin. Object-role modeling. *Encyclopedia of Database Systems*, pages 1941–1946, 2009.

[Han03] Uwe Hansmann. *Pervasive computing : The mobile world*. Springer Science & Business Media, 2003.

[HDPC02] Arran Holmes, Hakan Duman, and Anthony Pounds-Cornish. The idorm : Gateway to heterogeneous networking environments. In *International ITEA Workshop on Virtual Home Environments, Paderborn, Germany*, pages 20–21, 2002.

[Hen00] Thomas A Henzinger. *The theory of hybrid automata*. Springer, 2000.

[Hen03] Karen Henricksen. *A framework for context-aware pervasive computing applications*. University of Queensland Queensland, 2003.

[HG11] Espen Højsgaard and Arne J Glenstrup. The bpl tool : A tool for experimenting with bigraphical reactive systems. *Bigraphical Languages and their Simulation*, page 85, 2011.

[HGH+07] Jiang He, Tong Gao, Wei Hao, I-Ling Yen, and Farokh Bastani. A flexible content adaptation system using a rule-based approach. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1) :127–140, 2007.

[HI06] Karen Henricksen and Jadwiga Indulska. Developing context-aware pervasive computing applications : Models and approach. *Pervasive and mobile computing*, 2(1) :37–64, 2006.

[HM00] James Hendler and Deborah L McGuinness. The darpa agent markup language. *IEEE Intelligent systems*, 15(6) :67–73, 2000.

[HNBR97] Richard Hull, Philip Neaves, and James Bedford-Roberts. Towards situated computing. In *Wearable Computers, 1997. Digest of Papers., First International Symposium on*, pages 146–153. IEEE, 1997.

[Hoa69] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10) :576–580, 1969.

[Hop07] John E. Hopcroft. *Introduction to Automata Theory, Languages, and Computation*. Pearson Addison Wesley, 3rd edition, 2007.

[HPGMB11] José R Hoyos, Davy Preuveneers, Jesús J García-Molina, and Yolande Berbers. A dsl for context quality modeling in context-aware applications. In *Ambient Intelligence-Software and Applications*, pages 41–49. Springer, 2011.

[HSP+03] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann, and Werner Retschitzegger. Context-awareness on mobile devices-the hydrogen approach. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2003.

[Int02] Stephen S Intille. Designing a home of the future. *IEEE pervasive computing*, 1(2) :76–82, 2002.

[IS03] Jadwiga Indulska and Peter Sutton. Location management in pervasive systems. In *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003-Volume 21*, pages 143–151. Australian Computer Society, Inc., 2003.

[JB04] Gareth JF Jones and Peter J Brown. The role of context in information retrieval. In *ACM SIGIR 2004 Workshop on "Information Retrieval in Context"*, page 20, 2004.

[JCH+04] Xiaodong Jiang, Nicholas Y Chen, Jason I Hong, Kevin Wang, Leila Takayama, and James A Landay. *Siren : Context-aware computing for firefighting*. Springer, 2004.

[JM04] Ole Høgh Jensen and Robin Milner. Bigraphs and mobile processes (revised). Technical report, Technical Report UCAM-CL-TR-580, University of Cambridge, 2004.

[KA00] Mari Korkea-Aho. Context-aware applications survey. *Department of Computer Science, Helsinki University of Technology*, 2000.

[KBC02] Abdelmadjid Ketfi, Noureddine Belkhatir, and Pierre-Yves Cunin. Adaptation dynamique, concepts et experimentations. In *Proceedings of ICSSEA*, 2002.

[KKC+01] Lalana Kagal, Vlad Korolev, Harry Chen, Anupam Joshi, and Timothy Finin. Centaurus : A framework for intelligent services in a mobile environment. In *Distributed Computing Systems Workshop, 2001 International Conference on*, pages 195–201. IEEE, 2001.

[KL89] Michael Kifer and Georg Lausen. F-logic : a higher-order language for reasoning about objects, inheritance, and scheme. In *ACM SIGMOD Record*, volume 18, pages 134–146. ACM, 1989.

[KPVOGM05] Manuele Kirsch-Pinheiro, Marlène Villanova-Oliver, Jérôme Gensel, and Hervé Martin. Context-aware filtering for collaborative web systems : adapting the awareness information to the user's context. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 1668–1673. ACM, 2005.

[L+95] Henry Lieberman et al. Letizia : An agent that assists web browsing. *IJCAI (1)*, 1995 :924–929, 1995.

[LaC04] Laboratory for Context-dependent Mobile Communication LaCoMoCo. Bigraphical programming languages (bpl) @ONLINE, January 2004.

[LM00a] Gerard Lacey and Shane MacNamara. Context-aware shared control of a robot mobility aid for the elderly blind. *The International Journal of Robotics Research*, 19(11) :1054–1065, 2000.

[LM00b] James J Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In *CONCUR 2000 - Concurrency Theory*, pages 243–258. Springer, 2000.

[LRL+00] Gary T Leavens, Clyde Ruby, K Rustan M Leino, Erik Poll, and Bart Jacobs. Jml (poster session) : notations and tools supporting detailed design in java. In *Addendum to the 2000 proceedings of the conference on Object-oriented programming, systems, languages, and applications (Addendum)*, pages 105–106. ACM, 2000.

[LS00] Henry Lieberman and Ted Selker. Out of context : Computer systems that adapt to, and learn from, context. *IBM Systems Journal*, 39(3.4) :617–632, 2000.

[Mey92a] Bertrand Meyer. Applying'design by contract'. *Computer*, 25(10) :40–51, 1992.

[Mey92b] Bertrand Meyer. Eiffel-the language prentice hall. *Englewood Cliffs, NJ*, 1992.

[Mil80] Robin Milner. *A calculus of communicating systems*, volume 92. springer-Verlag Berlin, 1980.

[Mil93] Robin Milner. Action calculi, or syntactic action structures. In *Mathematical Foundations of Computer Science 1993*, pages 105–121. Springer, 1993.

[Mil96] Robin Milner. Calculi for interaction. *Acta Informatica*, 33(8) :707–737, 1996.

[Mil97] Robin Milner. *The definition of standard ML : revised*. MIT press, 1997.

[Mil99] Robin Milner. *Communicating and mobile systems : the pi calculus*. Cambridge university press, 1999.

[Mil01a] Robin Milner. Bigraphical reactive systems. In *CONCUR 2001-Concurrency Theory*, pages 16–35. Springer, 2001.

[Mil01b] Robin Milner. Bigraphical reactive systems : basic theory. Technical report, Technical Report 523, Computer Laboratory, University of Cambridge, 2001.

[Mil04] Robin Milner. Bigraphs for petri nets. In *Lectures on Concurrency and Petri Nets*, pages 686–701. Springer, 2004.

[Mil05] Robin Milner. Axioms for bigraphical structure. *Mathematical Structures in Computer Science*, 15(06) :1005–1032, 2005.

[Mil06] Robin Milner. Pure bigraphs : Structure and dynamics. *Information and computation*, 204(1) :60–122, 2006.

[Mil08] Robin Milner. Bigraphs and their algebra. *Electronic Notes in Theoretical Computer Science*, 209 :5–19, 2008.

[Mil09] Robin Milner. *The space and motion of communicating agents*. Cambridge University Press, 2009.

[Mou97] Alexandros Moukas. Amalthaea information discovery and filtering using a multiagent evolving ecosystem. *Applied Artificial Intelligence*, 11(5) :437–457, 1997.

[MPRB04] Ghita Kouadri Mostefaoui, Jacques Pasquier-Rocha, and Patrick Brezillon. Context-aware computing : a guide for the pervasive computing community. In *Pervasive Services, 2004. ICPS 2004. IEEE/ACS International Conference on*, pages 39–48. IEEE, 2004.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, i. *Information and computation*, 100(1) :1–40, 1992.

[MSKC04] Philip K McKinley, Seyed Masoud Sadjadi, Eric P Kasten, and Betty HC Cheng. Composing adaptive software. *Computer*, (7) :56–64, 2004.

[MVH+04] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10) :2004, 2004.

[Nis] Henning Niss. Formalizing geocast routing. Manuscript under preparation.

[NLW+12] EWT Ngai, TKP Leung, YH Wong, MCM Lee, PYF Chai, and YS Choi. Design and development of a context-aware decision support system for real-time accident handling in logistics. *Decision support systems*, 52(4) :816–827, 2012.

[OCL06] OMG OCL. Object constraint language. *OMG Specification, Version 2.0, formal/06-05*, 1, 2006.

[Omo91] Stephen M Omohundro. The sather language. Technical report, Technical report, International Computer Science Institute, Berkeley, Ca, 1991.

[Pas98] Jason Pascoe. Adding generic contextual capabilities to wearable computers. In *Wearable Computers, 1998. Digest of Papers. Second International Symposium on*, pages 92–99. IEEE, 1998.

[PDH12] Gian Perrone, Søren Debois, and Thomas T Hildebrandt. A model checker for bigraphs. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1320–1325. ACM, 2012.

[PKS12] Eloi Pereira, Christoph Kirsch, and Raja Sengupta. Biagents–a bigraphical agent model for structure-aware computation. *Cyber-Physical Cloud Computing Working Papers, CPCC Berkeley*, 2012.

[PNS+00] Daniela Petrelli, Elena Not, Carlo Strapparava, Oliviero Stock, and Massimo Zancanaro. Modeling context is like taking pictures. In *Proc. of the Workshop : The What, Who, Where, When, Why and How of Context-Awareness ? in CHI2000*, 2000.

[PRM98] Jason Pascoe, Nick S Ryan, and David R Morse. Human computer giraffe interaction : Hci in the field. 1998.

[Rai08] Claudia Raibulet. Facets of adaptivity. In *Software Architecture*, pages 342–345. Springer, 2008.

[RCDD98] Tom Rodden, Keith Cheverst, K Davies, and Alan Dix. Exploiting context in hci design for mobile systems. In *Workshop on human computer interaction with mobile devices*, pages 21–22. Citeseer, 1998.

[RDN06] MA Razzaque, Simon Dobson, and Paddy Nixon. Categorization and modeling of quality in context information. In *Proceedings of the IJCAI 2005 Workshop on AI and Autonomic Communications, 2005.© PLANET, FZI, ICCS, TUM, EPFL, CIM, INTRAINTL, LIPSZ, TRT, TXT Page 47 of*. Citeseer, 2006.

[Rey04] Dave Reynolds. Jena 2 inference support. *Online manual at http ://jena. sourceforge. net/inference/index. html*, 2004.

[Roh03] Eli Rohn. Predicting context aware computing performance. *Ubiquity*, 2003(February) :1–17, 2003.

[RPM98] Nick S Ryan, Jason Pascoe, and David R Morse. Enhanced reality fieldwork : the context-aware archaeological assistant. In *Computer applications in archaeology*. Tempus Reparatum, 1998.

[RPPM00] Michel Riveill, Marie-Claude Pellegrini, Olivier Potonniée, and Raphaël Marvie. Adaptabilité des applications pour des usagers mobiles. *OCM 2000*, 2000.

[Rua84] Lawrence M Ruane. Abstract data types in assembly language programming. *ACM SIGPLAN Notices*, 19(1) :63–67, 1984.

[Rya99] Nick Ryan. Contextml : Exchanging contextual information between a mobile client and the fieldnote server. *Computing Laboratory, University of Kent at Canterbury, http ://www. cs. kent. ac. uk/projects/mobicomp/fnc/ConteXtML. html*, 1999.

[SAP+09] Marjorie Skubic, Gregory Alexander, Mihail Popescu, Marilyn Rantz, and James Keller. A smart home application to eldercare : Current status and lessons learned. *Technology and Health Care*, 17(3) :183–201, 2009.

[Sar89] P Sarrat. Le projet français habitat intelligent/domotique, 1989.

[SAW94] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE, 1994.

[SB05] Quan Z Sheng and Boualem Benatallah. Contextuml : a uml-based modeling language for model-driven development of context-aware web services. In *Mobile Business, 2005. ICMB 2005. International Conference on*, pages 206–212. IEEE, 2005.

[SBG99] Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Computers & Graphics*, 23(6) :893–901, 1999.

[SC01] Nary Subramanian and Lawrence Chung. Software architecture adaptability : an nfr approach. In *Proceedings of the 4th International Workshop on Principles of Software Evolution*, pages 52–61. ACM, 2001.

[SDA98] Daniel Salber, Anind K Dey, and Gregory D Abowd. Ubiquitous computing : Defining an hci research agenda for an emerging interaction paradigm. 1998.

[Sev12] Michele Sevegnani. *Bigraphs with sharing and applications in wireless networks*. PhD thesis, University of Glasgow, 2012.

[Sew02] Peter Sewell. From rewrite rules to bisimulation congruences. *Theoretical Computer Science*, 274(1) :183–230, 2002.

[SLP03] Thomas Strang and Claudia Linnhoff-Popien. Service interoperability on context level in ubiquitous computing environments. In *Intl. Conf. on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR2003w)*, 2003.

[SLP04] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *Workshop Proceedings*, 2004.

[SLPF03] Thomas Strang, Claudia Linnhoff-Popien, and Korbinian Frank. Cool : A context ontology language to enable contextual interoperability. In *Distributed applications and interoperable systems*, pages 236–247. Springer, 2003.

[SMLP02] Michael Samulowitz, Florian Michahelles, and Claudia Linnhoff-Popien. Capeus : An architecture for context-aware selection and execution of services. In *New developments in distributed applications and interoperable systems*, pages 23–39. Springer, 2002.

[SUC10] Michele Sevegnani, Chris Unsworth, and Muffy Calder. A sat based algorithm for the matching problem in bigraphs with sharing. *University of Glasgow, Tech. Rep*, 2010.

[SVL01] Albrecht Schmidt and Kristof Van Laerhoven. How to build smart appliances ? *Personal Communications, IEEE*, 8(4) :66–71, 2001.

[Vol98] Martin Volk. The automatic translation of idioms. machine translation vs. translation memory systems. *Machine Translation : Theory, Applications, and Evaluation, An Assessment of the State-of-the-art, St. Augustin, Gardez Verlag*, 1998.

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3) :94–104, 1991.

[Wei93] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7) :75–84, 1993.

[Wei94] Marc Weiser. The world is not a desktop. *interactions*, 1(1) :7–8, 1994.

[Wei95] Mark Weiser. Human-computer interaction. chapter the computer for the 21st century, 1995.

[WJH97] Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the active office. *Personal Communications, IEEE*, 4(5) :42–47, 1997.

[WSK08] Peter Wolf, Andreas Schmidt, and Michael Klein. Soprano-an extensible, open aal platform for elderly people based on semantical contracts. In *3rd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI'08), 18th European Conference on Artificial Intelligence (ECAI 08), Patras, Greece*. Citeseer, 2008.

[WXL11] Ju-Shu Wang, Dong Xu, and Zhou Lei. Formalizing the structure and behaviour of context-aware systems in bigraphs. In *Software and Network Engineering (SSNE), 2011 First ACIS International Symposium on*, pages 89–94. IEEE, 2011.

[WZGP04] Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology based context modeling and reasoning using owl. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 18–22. Ieee, 2004.

[XC00] Jinxi Xu and W Bruce Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems (TOIS)*, 18(1) :79–112, 2000.

[Xia03] Wang Xiaohang. The context gateway : A pervasive computing infrastructure for context aware service. *Report submitted to School of Computing, National University of Singapore & Context-Aware Department., Institute for Infocomm Research*, 2003.

[XXL11] De-Zhen Xu, Dong Xu, and Zhou Lei. Bigraphical model of context-aware in ubiquitous computing environments. In *Services Computing Conference (APSCC), 2011 IEEE Asia-Pacific*, pages 389–394. IEEE, 2011.

[ZBT+09] Nadia Zouba, François Brémond, Monique Thonnat, Alain Anfosso, Eric Pascual, Patrick Mallea, Veronique Mailland, and Olivier Guerin. A computer system to monitor older adults at home : Preliminary results. In *Gerontechnology Journal*, volume 8, pages 129–139, 2009.

[ZEBD98] Eli Zelkha, Brian Epstein, S Birrell, and C Dodsworth. From devices to ambient intelligence. In *Digital living room conference*, volume 6, 1998.

[ZLO07] Andreas Zimmermann, Andreas Lorenz, and Reinhard Oppermann. An operational definition of context. In *Modeling and using context*, pages 558–571. Springer, 2007.

Annexe A : BigCAS-Tool

A.1 Fichier plugin.xml de BigCAS-Tool

Listing A.1– Fichier plugin.xml de BigCAS-Tool

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
   <extension-point id="org.bigcas_tool.interactionManagers" name="Interaction
     managers" schema="expoints/org.bigcas_tool.interactionManagers.exsd"/>

   <extension
         id="application"
         point="org.eclipse.core.runtime.applications">
      <application>
         <run
               class="org.bigcas_tool.application.standalone.
  BigCASToolApplication">
         </run>
      </application>
   </extension>
   <extension
         point="org.eclipse.ui.perspectives">
      <perspective
            class="org.bigcas_tool.application.plugin.BigCASToolPerspective"
            icon="resources/icons/icon32.png"
            id="org.bigcas_tool.perspective"
            name="BigCAS-Tool">
      </perspective>
   </extension>
   <extension
         point="org.eclipse.ui.perspectiveExtensions">
      <perspectiveExtension
            targetID="*">
         <newWizardShortcut
               id="org.bigcas_tool.NewAgentWizard">
         </newWizardShortcut>
         <newWizardShortcut
               id="org.bigcas_tool.NewBRSWizard">
         </newWizardShortcut>
         <newWizardShortcut
               id="org.bigcas_tool.NewRuleWizard">
```

```
36          </newWizardShortcut>
37          <newWizardShortcut
38                id="org.bigcas_tool.NewSignatureWizard">
39          </newWizardShortcut>
40          <newWizardShortcut
41                id="org.bigcas_tool.NewOperationWizard">
42          </newWizardShortcut>
43       </perspectiveExtension>
44    </extension>
45    <extension
46          id="product"
47          point="org.eclipse.core.runtime.products">
48       <product
49             application="org.bigcas_tool.application"
50             description="A Bigraphical Environment for Modeling Context-Aware
      Systems"
51             name="BigCAS-Tool">
52          <property
53                name="windowImages"
54                value="resources/icons/icon16.png,resources/icons/icon32.png">
55          </property>
56          <property
57                name="appName"
58                value="BigCAS-Tool">
59          </property>
60          <property
61                name="aboutImage"
62                value="resources/logo-help.png">
63          </property>
64          <property
65                name="preferenceCustomization"
66                value="plugin_customization.ini">
67          </property>
68          <property
69                name="startupProgressRect"
70                value="2,284,451,10">
71          </property>
72          <property
73                name="startupForegroundColor"
74                value="000000">
75          </property>
76          <property
77                name="startupMessageRect"
78                value="22,264,411,20">
79          </property>
80          <property
81                name="aboutText"
82                value="BigCAS-Tool: A Bigraphical Environment for Modeling
      Context-Aware Systems
83                Version: 1.0.0
84                (c) Copyright Taha Abdelmoutaleb Cherfia 2014.  All rights
                    reserved.
85                Visit http://github.com/cherfia/BigCAS-Tool
86
87                BigCAS-Tol is build on Big Red "the graphical development
```

```
                environment for bigraphs"
88              Copyright (c) 2009, 2010 Alexander Faithfull. All rights
89              reserved.
90
91              BigCAS-Tool is built on the Eclipse Rich Client Platform.
92              Copyright (c) Eclipse contributors and others 2000,2014.
93              All rights reserved. Visit http://eclipse.org/
94
95              This product includes software developed by the
96              Apache Software Foundation, http://apache.org/">
97        </property>
98     </product>
99  </extension>
100  <extension
101        point="org.eclipse.ui.editors">
102     <editor
103           class="org.bigcas_tool.editors.bigraph.BigraphEditor"
104           contributorClass="org.bigcas_tool.editors.bigraph.
    BigraphEditorActionBarContributor"
105           default="true"
106           extensions="bigraph, bigraph-agent"
107           icon="resources/icons/icon16.png"
108           id="org.bigcas_tool.BigraphEditor"
109           name="Context-Aware Bigraph Editor">
110        <contentTypeBinding
111              contentTypeId="org.bigcas_tool.bigraph">
112        </contentTypeBinding>
113     </editor>
114     <editor
115           class="org.bigcas_tool.editors.signature.SignatureEditor"
116           contributorClass="org.bigcas_tool.editors.signature.
    ActionBarContributor"
117           default="true"
118           extensions="bigraph-signature"
119           icon="resources/icons/signature.png"
120           id="org.bigcas_tool.SignatureEditor"
121           name="Context-Aware Signature Editor">
122        <contentTypeBinding
123              contentTypeId="org.bigcas_tool.signature">
124        </contentTypeBinding>
125     </editor>
126     <editor
127           class="org.bigcas_tool.editors.rule.RuleEditor"
128           contributorClass="org.bigcas_tool.editors.rule.ActionBarContributor"
129           default="true"
130           extensions="bigraph-rule"
131           icon="resources/icons/reaction-rule.png"
132           id="org.bigcas_tool.RuleEditor"
133           name="Context-Aware Reaction Rule Editor">
134        <contentTypeBinding
135              contentTypeId="org.bigcas_tool.rule">
136        </contentTypeBinding>
137     </editor>
138     <editor
139           class="org.bigcas_tool.editors.simulation_spec.SimulationSpecEditor"
```

```
140             contributorClass="org.bigcas_tool.editors.simulation_spec.
    ActionBarContributor"
141             default="true"
142             extensions="bigraph-simulation-spec"
143             icon="resources/icons/simulation.png"
144             id="org.bigcas_tool.SimulationSpecEditor"
145             name="Simulation Spec Editor">
146          <contentTypeBinding
147                contentTypeId="org.bigcas_tool.simulation_spec">
148          </contentTypeBinding>
149       </editor>
150    </extension>
151    <extension
152          point="org.eclipse.ui.exportWizards">
153       <category
154             id="org.bigcas_tool.export"
155             name="BigCAS-Tool">
156       </category>
157       <wizard
158             category="org.bigcas_tool.export"
159             class="org.bigcas_tool.wizards.export.TextExportWizard"
160             icon="resources/icons/wizards/text.png"
161             id="BigCAS-Tool.TikZExportWizard"
162             name="Export as text">
163          <description>
164             Export a BigCAS-Tool document to a text format.
165          </description>
166       </wizard>
167    </extension>
168    <extension
169          point="org.eclipse.ui.importWizards">
170       <category
171             id="org.bigcas_tool.import"
172             name="BigCAS-Tool">
173       </category>
174    </extension>
175    <extension
176          point="org.eclipse.ui.newWizards">
177       <category
178             id="org.bigcas_tool.new"
179             name="BigCAS-Tool">
180       </category>
181       <wizard
182             category="org.bigcas_tool.new"
183             class="org.bigcas_tool.wizards.creation.NewBRSWizard"
184             finalPerspective="org.bigcas_tool.perspective"
185             icon="resources/icons/icon16.png"
186             id="org.bigcas_tool.NewBRSWizard"
187             name="Context-Aware Bigraphical Reactive System"
188             project="true">
189          <description>
190             Create a new context-aware bigraphical reactive system.
191          </description>
192       </wizard>
193       <wizard
```

```
194             category="org.bigcas_tool.new"
195             class="org.bigcas_tool.wizards.creation.NewAgentWizard"
196             icon="resources/icons/icon16.png"
197             id="org.bigcas_tool.NewAgentWizard"
198             name="Context-Aware Bigraph"
199             project="false">
200          <description>
201             Create a new context-aware bigraph in an existing bigraphical
                  reactive system.
202          </description>
203       </wizard>
204       <wizard
205             category="org.bigcas_tool.new"
206             class="org.bigcas_tool.wizards.creation.NewAgentWizard"
207             icon="resources/icons/icon32.png"
208             id="org.bigcas_tool.NewContextUnawareWizard"
209             name="Context-Unaware Bigraph"
210             project="false">
211          <description>
212             Create a new context-unaware bigraph in an existing bigraphical
                  reactive system.
213          </description>
214       </wizard>
215       <wizard
216             category="org.bigcas_tool.new"
217             class="org.bigcas_tool.wizards.creation.NewSignatureWizard"
218             icon="resources/icons/signature.png"
219             id="org.bigcas_tool.NewSignatureWizard"
220             name="Signature"
221             project="false">
222          <description>
223             Create a new context-aware signature in an existing bigraphical
                  reactive system.
224          </description>
225       </wizard>
226       <wizard
227             category="org.bigcas_tool.new"
228             class="org.bigcas_tool.wizards.creation.NewRuleWizard"
229             icon="resources/icons/reaction-rule.png"
230             id="org.bigcas_tool.NewRuleWizard"
231             name="Reaction Rule"
232             project="false">
233          <description>
234             Create a new context-aware reaction rule in an existing bigraphical
                  reactive system.
235          </description>
236       </wizard>
237       <wizard
238             category="org.bigcas_tool.new"
239             class="org.bigcas_tool.wizards.creation.NewSimulationSpecWizard"
240             icon="resources/icons/simulation.png"
241             id="org.bigcas_tool.NewSimulationSpecWizard"
242             name="Simulation spec"
243             project="false">
244          <description>
```

```
245             Create a new simulation spec in an existing bigraphical reactive
                  system.
246          </description>
247       </wizard>
248    </extension>
249    <extension
250          point="org.eclipse.core.contenttype.contentTypes">
251       <content-type
252             base-type="org.eclipse.core.runtime.xml"
253             file-extensions="bigraph, bigraph-agent"
254             id="org.bigcas_tool.bigraph"
255             name="Context-Aware Bigraph"
256             priority="normal">
257       </content-type>
258       <content-type
259             base-type="org.eclipse.core.runtime.xml"
260             file-extensions="bigraph-rule"
261             id="org.bigcas_tool.rule"
262             name="Context-Aware Reaction Rule"
263             priority="normal">
264       </content-type>
265       <content-type
266             base-type="org.eclipse.core.runtime.xml"
267             file-extensions="bigraph-signature"
268             id="org.bigcas_tool.signature"
269             name="Context-Aware Signature"
270             priority="normal">
271       </content-type>
272       <content-type
273             base-type="org.eclipse.core.runtime.xml"
274             file-extensions="bigraph-simulation-spec"
275             id="org.bigcas_tool.simulation_spec"
276             name="Simulation spec"
277             priority="normal">
278       </content-type>
279       <content-type
280             base-type="org.eclipse.core.runtime.xml"
281             file-extensions="bigraph, bigraph-operation"
282             id="org.bigcas_tool.operation"
283             name="Operation"
284             priority="normal">
285       </content-type>
286    </extension>
287    <extension
288          point="org.bigraph.model.wrapper.import">
289       <importer
290             class="org.bigraph.model.loaders.BigraphXMLLoader"
291             contentType="org.bigcas_tool.bigraph"
292             icon="resources/icons/icon16.png"
293             id="big_red.importer1"
294             name="Bigraph from XML">
295       </importer>
296       <importer
297             class="org.bigraph.model.loaders.SignatureXMLLoader"
298             contentType="org.bigcas_tool.signature"
```

```
299             icon="resources/icons/signature.png"
300             id="big_red.importer2"
301             name="Signature from XML">
302       </importer>
303       <importer
304             class="org.bigraph.model.loaders.ReactionRuleXMLLoader"
305             contentType="org.bigcas_tool.rule"
306             icon="resources/icons/reaction-rule.png"
307             id="big_red.importer3"
308             name="Reaction rule from XML">
309       </importer>
310       <importer
311             class="org.bigraph.model.loaders.SimulationSpecXMLLoader"
312             contentType="org.bigcas_tool.simulation_spec"
313             icon="resources/icons/simulation.png"
314             id="big_red.importer4"
315             name="Simulation spec from XML">
316       </importer>
317       <participant
318             class="org.bigcas_tool.model.load_save.BigCASToolXMLUndecorator"
319             id="RedUndecorator" />
320       <participant
321             class="org.bigcas_tool.model.load_save.AliasSupport$Undecorator"
322             id="AliasUndecorator" />
323       <participant
324             class="org.bigcas_tool.model.LinkStyleUtilities$Undecorator"
325             id="LinkStyleUndecorator">
326       </participant>
327       <participant
328             class="org.bigcas_tool.model.load_save.
    BigCASToolXMLEdits$LoadParticipant"
329             id="RedXMLEditLoader">
330          <enablement>
331             <instanceof
332                   value="org.bigraph.model.loaders.EditXMLLoader">
333             </instanceof>
334          </enablement>
335       </participant>
336       <participant
337             class="org.bigcas_tool.model.load_save.loaders.
    ChangeCompatibilityLoader"
338             id="ChangeCompatibilityLoader">
339          <enablement>
340             <instanceof
341                   value="org.bigraph.model.loaders.ReactionRuleXMLLoader">
342             </instanceof>
343          </enablement>
344       </participant>
345    </extension>
346    <extension
347          point="org.bigraph.model.wrapper.export">
348       <exporter
349             class="org.bigcas_tool.model.load_save.savers.BigraphTikZSaver"
350             exports="org.bigraph.model.Bigraph"
351             icon="resources/icons/wizards/vector.png"
```

```
352             id="bigcas-tool.TikZexporter"
353             name="TikZ image">
354          <description>
355             Export a bigraph as a TikZ image, suitable for use in papers or
                  high-resolution printing.
356          </description>
357       </exporter>
358       <exporter
359             class="org.bigraph.model.savers.BigraphXMLSaver"
360             contentType="org.bigcas_tool.bigraph"
361             exports="org.bigraph.model.Bigraph"
362             icon="resources/icons/icon16.png"
363             id="bigcas-tool.BigraphExporter"
364             name="Bigraph as XML">
365          <description>
366             Export a bigraph as a XML document, equivalent to that produced by
                  File -> Save.
367          </description>
368       </exporter>
369       <exporter
370             class="org.bigraph.model.savers.SignatureXMLSaver"
371             contentType="org.bigcas_tool.signature"
372             exports="org.bigraph.model.Signature"
373             icon="resources/icons/signature.png"
374             id="bigcas-tool.SignatureExporter"
375             name="Signature as XML">
376       </exporter>
377       <exporter
378             class="org.bigraph.model.savers.SimulationSpecXMLSaver"
379             contentType="org.bigcas_tool.simulation_spec"
380             exports="org.bigraph.model.SimulationSpec"
381             icon="resources/icons/simulation.png"
382             id="bigcas-tool.SimulationExporter"
383             name="Simulation spec as XML">
384       </exporter>
385       <exporter
386             class="org.bigraph.model.savers.ReactionRuleXMLSaver"
387             contentType="org.bigcas_tool.rule"
388             exports="org.bigraph.model.ReactionRule"
389             icon="resources/icons/reaction-rule.png"
390             id="bigcas-tool.ReactionRuleExporter"
391             name="Reaction rule as XML">
392       </exporter>
393       <exporter
394             class="org.bigcas_tool.model.load_save.savers.BigraphBPLToolSaver"
395             exports="org.bigraph.model.Bigraph"
396             id="bigcas-tool.BPLExporter"
397             name="BPL Tool">
398       </exporter>
399       <exporter
400             class="org.bigcas_tool.model.load_save.savers.BigraphTraverseSaver"
401             exports="org.bigraph.model.Bigraph"
402             icon="resources/icons/wizards/text.png"
403             id="bigcas-tool.TextExporter"
404             name="Traverse">
```

```
405       </exporter>
406       <participant
407             class="org.bigcas_tool.model.load_save.BigCASToolXMLDecorator"
408             id="RedDecorator" />
409       <participant
410             class="org.bigcas_tool.model.load_save.AliasSupport$Decorator"
411             id="AliasDecorator" />
412       <participant
413             class="org.bigcas_tool.model.LinkStyleUtilities$Decorator"
414             id="LinkStyleDecorator">
415       </participant>
416       <participant
417             class="org.bigcas_tool.model.load_save.
    BigCASToolXMLEdits$SaveParticipant"
418             id="RedXMLEditSaver">
419          <enablement>
420             <instanceof
421                   value="org.bigraph.model.savers.EditXMLSaver">
422             </instanceof>
423          </enablement>
424       </participant>
425    </extension>
426    <extension
427          point="org.eclipse.ui.preferencePages">
428       <page
429             class="org.bigcas_tool.preferences.BigCASToolPreferencePage"
430             id="org.bigcas_tool.preferences.RedPreferencePage"
431             name="BigCAS-Tool">
432       </page>
433    </extension>
434    <extension
435          point="org.eclipse.core.runtime.preferences">
436       <initializer
437             class="org.bigcas_tool.preferences.BigCASToolPreferences">
438       </initializer>
439    </extension>
440    <extension
441          id="org.bigcas_tool.utilities.resources.builder.BigraphBuilder"
442          name="BigCAS-Tool bigraph builder"
443          point="org.eclipse.core.resources.builders">
444       <builder
445             hasNature="true">
446          <run
447                class="org.bigcas_tool.builder.BigraphBuilder">
448          </run>
449       </builder>
450    </extension>
451    <extension
452          id="org.bigcas_tool.utilities.resources.builder.BigraphNature"
453          name="BigCAS-Tool bigraph nature"
454          point="org.eclipse.core.resources.natures">
455       <runtime>
456          <run
457                class="org.bigcas_tool.builder.BigraphNature">
458          </run>
```

```
459       </runtime>
460       <builder
461             id="org.bigcas_tool.utilities.resources.builder.BigraphBuilder">
462       </builder>
463    </extension>
464    <extension
465          point="org.eclipse.ltk.core.refactoring.moveParticipants">
466       <moveParticipant
467             class="org.bigcas_tool.refactoring.MoveParticipant"
468             id="BigCAS-Tool.moveParticipant2"
469             name="Bigraph nature move participant">
470          <enablement>
471             <with
472                   variable="affectedNatures">
473                <iterate
474                      ifEmpty="false"
475                      operator="or">
476                   <equals
477                         value="org.bigcas_tool.utilities.resources.builder.
  BigraphNature">
478                   </equals>
479                </iterate>
480             </with>
481          </enablement>
482       </moveParticipant>
483    </extension>
484 </plugin>
```

A.2 Point d'extension org.bigcas_tool.interactionManagers

Listing A.2– Point d'extension org.bigcas_tool.interactionManagers

```
1     <?xml version='1.0' encoding='UTF-8'?>
2 <!-- Schema file written by PDE -->
3 <schema targetNamespace="org.bigcas_tool" xmlns="http://www.w3.org/2001/XMLSchema
    ">
4 <annotation>
5       <appinfo>
6          <meta.schema plugin="org.bigcas_tool" id="org.bigcas_tool.
             interactionManagers" name="Interaction managers"/>
7       </appinfo>
8       <documentation>
9          The interaction managers extension point allows plug-ins to contribute
             interaction managers, classes responsible for the interaction between
             BigCAS-Tool and external tools.
10      </documentation>
11    </annotation>
12
13    <element name="extension">
14       <annotation>
15          <appinfo>
16             <meta.element />
17          </appinfo>
18       </annotation>
```

```
19        <complexType>
20           <sequence>
21              <element ref="interactionManager" minOccurs="0" maxOccurs="unbounded"
                  />
22           </sequence>
23           <attribute name="point" type="string" use="required">
24              <annotation>
25                 <documentation>
26
27                 </documentation>
28              </annotation>
29           </attribute>
30           <attribute name="id" type="string">
31              <annotation>
32                 <documentation>
33
34                 </documentation>
35              </annotation>
36           </attribute>
37           <attribute name="name" type="string">
38              <annotation>
39                 <documentation>
40
41                 </documentation>
42                 <appinfo>
43                    <meta.attribute translatable="true"/>
44                 </appinfo>
45              </annotation>
46           </attribute>
47        </complexType>
48     </element>
49
50     <element name="interactionManager">
51        <complexType>
52           <sequence>
53              <element ref="description" minOccurs="0" maxOccurs="1"/>
54           </sequence>
55           <attribute name="id" type="string" use="required">
56              <annotation>
57                 <documentation>
58                    A unique name that will be used to identify this interaction
                      manager.
59                 </documentation>
60              </annotation>
61           </attribute>
62           <attribute name="name" type="string" use="required">
63              <annotation>
64                 <documentation>
65                    A translatable name used to represent this interaction manager
                      in the user interface.
66                 </documentation>
67                 <appinfo>
68                    <meta.attribute translatable="true"/>
69                 </appinfo>
70              </annotation>
```

```
71          </attribute>
72          <attribute name="class" type="string" use="required">
73             <annotation>
74                <documentation>
75                   The class for this interaction manager (implementing dk.itu.
                       big_red.tools.IInteractionManager). The class must have a
                       nullary constructor.
76                </documentation>
77                <appinfo>
78                   <meta.attribute kind="java" basedOn=":dk.itu.big_red.tools.
                       IInteractionManager"/>
79                </appinfo>
80             </annotation>
81          </attribute>
82          <attribute name="icon" type="string">
83             <annotation>
84                <documentation>
85                   A relative path to an icon that will be used in the user
                       interface.
86                </documentation>
87                <appinfo>
88                   <meta.attribute kind="resource"/>
89                </appinfo>
90             </annotation>
91          </attribute>
92       </complexType>
93    </element>
94
95    <element name="description" type="string">
96       <annotation>
97          <appinfo>
98             <meta.element translatable="true"/>
99          </appinfo>
100         <documentation>
101            An optional sub-element whose body should describe this interaction
                 manager.
102         </documentation>
103      </annotation>
104   </element>
105
106   <annotation>
107      <appinfo>
108         <meta.section type="since"/>
109      </appinfo>
110      <documentation>
111         2.1
112      </documentation>
113   </annotation>
114</schema>
```

Annexe B : Point d'extension BigMC

B.1 Fichier plugin.xml de BigMC

Listing B.1– Fichier plugin.xml de BigMC

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
   <extension
         point="org.bigcas_tool.interactionManagers">
      <interactionManager
            class="org.bigraph.bigmc.bigcas_tool.BigMCInteractionManager"
            id="BigMC for BigCAS-Tool.interactionManager1"
            name="BigMC user interface 2.0">
      </interactionManager>
   </extension>
   <extension
         point="org.bigraph.model.wrapper.export">
      <exporter
            class="org.bigraph.bigmc.bigcas_tool.SimulationSpecBigMCSaver"
            exports="org.bigraph.model.SimulationSpec"
            id="BigMC for BigCAS-Tool.exporter1"
            name="BigMC term language">
      </exporter>
   </extension>
   <extension
         point="org.eclipse.core.runtime.preferences">
      <initializer
            class="org.bigraph.bigmc.bigcas_tool.Preferences">
      </initializer>
   </extension>
   <extension
         point="org.eclipse.ui.preferencePages">
      <page
            category="org.bigcas_tool.preferences.RedPreferencePage"
            class="org.bigraph.bigmc.bigcas_tool.PreferencePage"
            id="org.bigraph.bigmc.red.PreferencePage"
            name="BigMC for BigCAS-Tool">
      </page>
   </extension>
   <extension
         point="org.eclipse.ui.importWizards">
```

```
38      <wizard
39            category="org.bigcas_tool.import"
40            class="org.bigraph.bigmc.bigcas_tool.bgm.ImportWizard"
41            id="org.bigraph.bigmc.red.importWizards.ImportWizard"
42            name="Import BigMC file">
43         <description>
44            Import a BigMC file as a new BigCAS-Tool project.
45         </description>
46      </wizard>
47   </extension>
48 </plugin>
```